Premios para los lectores • Estrategia • C y Ensamblador • Aventura • DIV 2 • Arcade
La primera revista dedicada a aprender a programar videojuegos
DIV manía
995 ptas.
PORTUGAL 990 ESC (CONT) 5,98 €
www.prensatecnica.com
Año 1 • Número 2
•DIV 2
Las principales caracerísticas del mejor entorno de trabajo
•Dibujo y diseño
El más completo curso de diseño de juegos
•DIV Developer
Los programas ganadores del concurso de este número
•Work in progress
Necro: el nuevo proyecto de los desarrolladores de Island Dream
•Programa tus juegos
Todos los secretos de cada género
•DIV interno
Aspectos avanzados
CONCURSO JUEGOS DIV
Prens@ Técnic@
8 413042 087533 00002
Cómo hacerte rico programando juegos

Tu PC siempre a punto con PC DRIVER

Tarjetas gráficas, impresoras, Bios, unidades Zip, discos duros, placas base, memorias, escáneres, monitores, digitalizadoras, módems, coprocesadores , joysticks, tarjetas de sonido...

¿Qué contiene PC Driver?

Más de 1.400 drivers

Alrededor de 900 horas de búsqueda en Internet

Con este número te ahorrarás más de 100.000 ptas. en gastos de Internet y teléfono



¿Qué es PC Driver?



***PC DRIVER** es la revista que irrumpe en el mercado para hacer más fácil la vida del aficionado a la informática.*

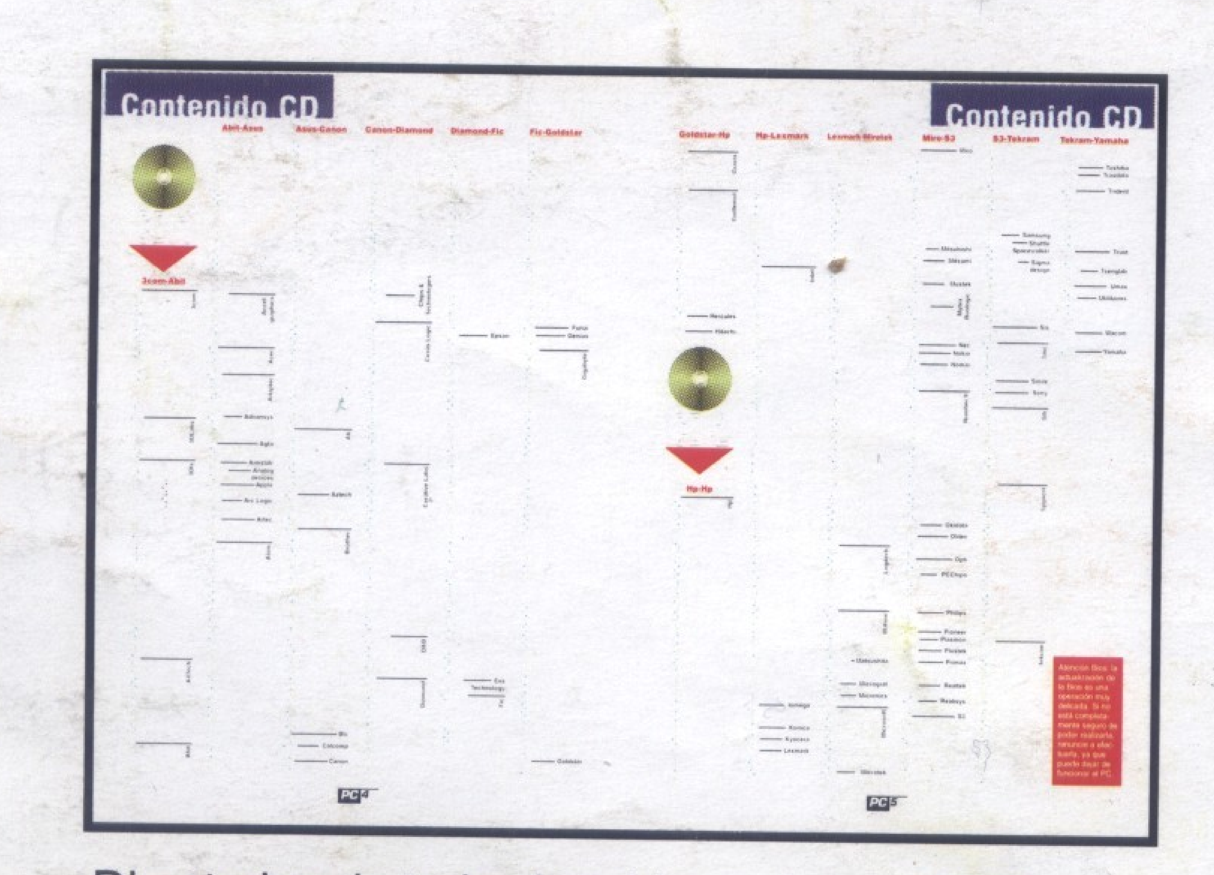
Todo el mundo ha experimentado la dificultad que supone conseguir un determinado driver o, simplemente, mantenerlos juntos y ordenados.

***PC DRIVER** contiene en sus 2 CD ROM el compendio de todos los drivers, modelos y sistemas, actualizaciones y viejas versiones de las marcas más distribuidas por todos los fabricantes de Hardware, para tener tu PC siempre a punto.*

¿Sabes resolver el Puzzle de la Informática?



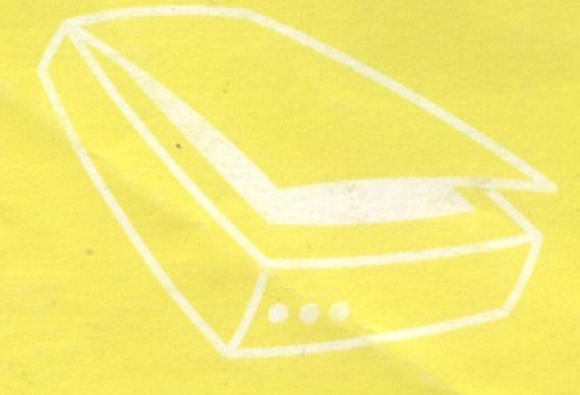
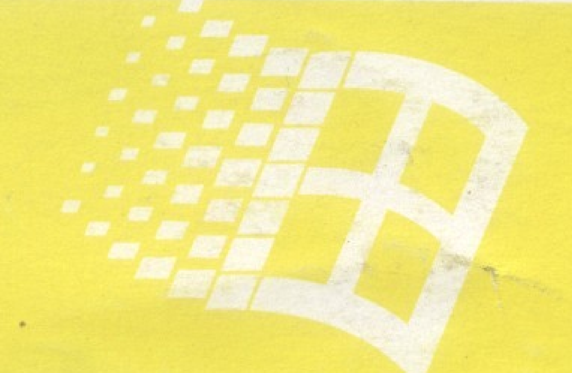
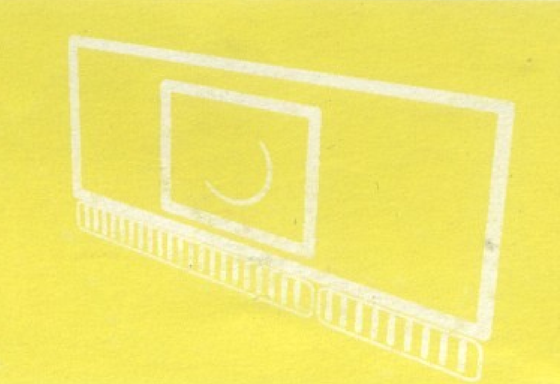
Los primeros pasos para actualizar nuestro PC.



Directorios de todos los drivers de los CD-Roms.



Análisis de los puntos críticos de la placa base.





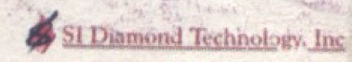










LA REVISTA IMPRESCINDIBLE PARA QUE FUNCIONE TU PC



La revista imprescindible para el mantenimiento de tu PC

PC DRIVER

La revista que contiene todos los drivers del mercado
www.prensatecnica.com
Año 1 • Número 1 • 995 ptas. • 5,98 €

Todos los drivers del mercado

3Com • 3D Labs • 3DFX • A4Tech • Abit • AccelGraphics • Accton • Acer • Adaptec • Advansys • Agfa • Aimslab • Analog Devices • Apple • Ark Logic • Artec • Asus • Ati • Aztech • Brother • Btc • Calcomp • Canon • Chips & Technologies • Cirrus Logic • Citizen • Creative Labs • D&B • Diamond • Epson • Ess Technology • Fic • Funai • Genius • Gigabyte • Goldstar • Gravis • Guillemot • Hercules • Hitachi • Hewlett-Packard • Intel • Iomega • Konica • Kyocera • Lexmark • Logitech • Matrox • Matsushita • Panasonic • Micrograf • Micronics • Microsoft • Mirotek • Miro • Miro (Pinnacle System) • Mitsubishi • Mitsumi • Mustek • Mylex • BusLogic • Nec • Neomagic • Nokia • Nomai • Number 9 • Okidata • Olitec • Opti • PC Chips • Philips • Pioneer • Plasmon • Plustek • Primax • Realtek • Relisys • Ricoh • S3 • Samsung • Sanyo • Shuttle Spacewalker • Sigmadesigns • Sis • Smc • Smile • Sony • Stb • Syquest • Teac • Tekram • Texas Instrument • Toshiba • Traxdata • Trident • Trust • Tseng Labs • Umax • Wacom • Yamaha

Impresoras
Incluimos todos los drivers necesarios para no tener que desechar tus viejos periféricos

Escáneres
El driver que esperabas para poner al día tu escáner

Tarjetas de vídeo
Saca el máximo partido a tu tarjeta de vídeo

La Bios
Ten siempre a mano la última versión operativa de las Bios

Para todos
Compatibles con Windows 95, 98 y NT

Número 1
995 ptas.

Conoce cómo funciona tu PC a nivel interno

Analizamos todos los componentes de tu PC

Principiantes
Aprenda a dar los primeros pasos para realizar instalaciones y actualizaciones

Todos los drivers
Una recopilación de todos los drivers de los componentes de nuestro PC.

Todos los sistemas
Drivers para Windows 95/98 y NT.

Útil
Consiga en dos CD-Roms más de 1.400 drivers.

Práctico
Drivers y controladores organizados para localizarlos de manera fácil y rápida

Actual
Las más recientes versiones y actualizaciones.

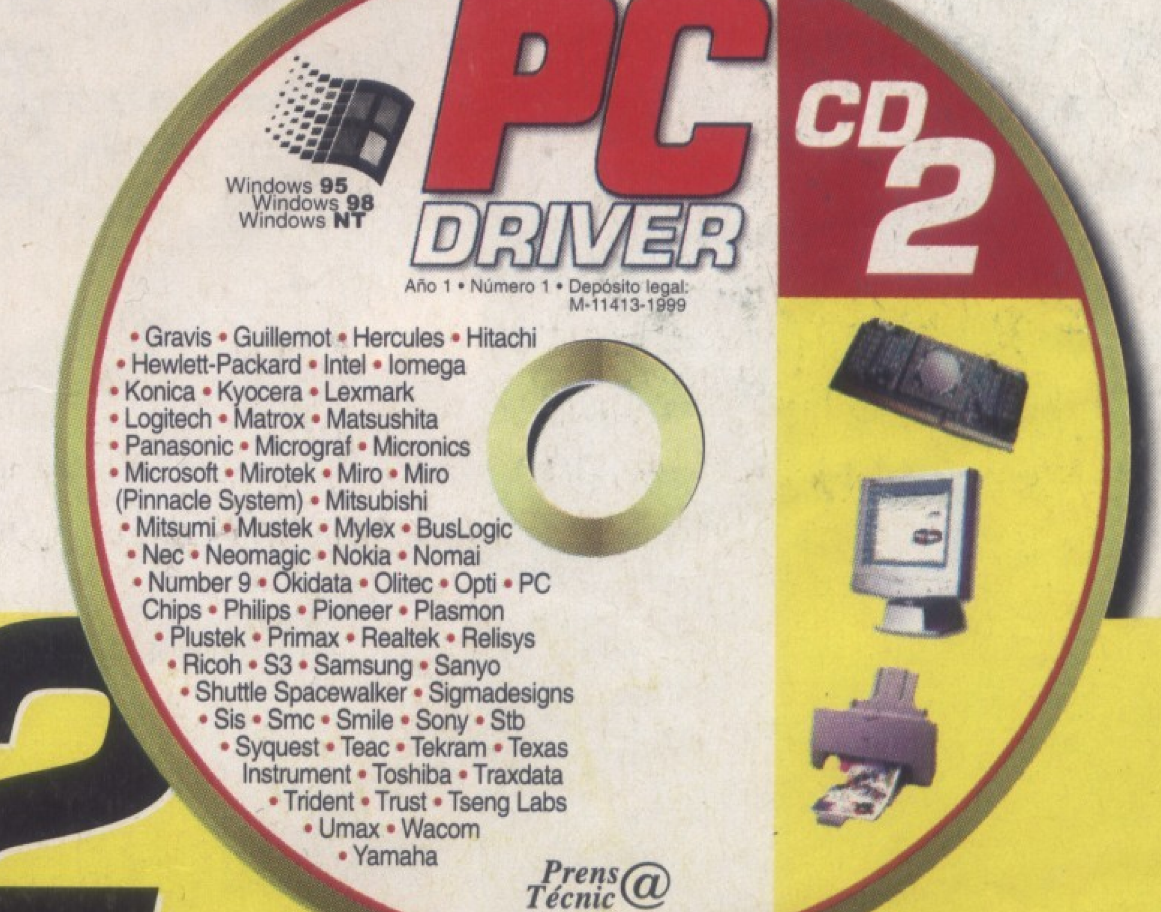
995 ptas. • 5,98 €

Con 2 CD-Roms

Cada 2 meses en tu quiosco

Ten siempre a mano los CD-Rom de PC DRIVER

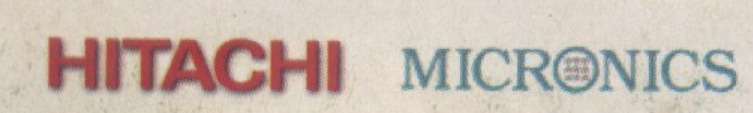


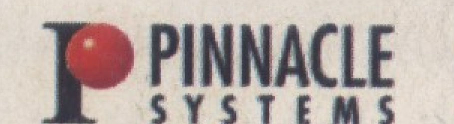


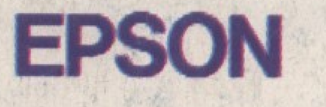
2











EPSON











¿Amas la programación de juegos...? Léete estas líneas

Grande ha sido la expectación levantada frente a este segundo número de la revista DIV Manía, y nosotros hemos agradecido el apremio de los lectores, pues eso indicaba que tenían ganas de hincarle el diente a lo que les ofrecíamos. Debemos disculparnos ante ellos por nuestra tardanza y, sin embargo, lo cierto es que se debe a nuestro deseo de ofreceros un producto mucho mejor. La principal causa del retraso es el propio DIV 2, que nosotros queríamos ofrecer en exclusiva a todos nuestros lectores. La aparición de DIV 2 es un hecho, y hemos logrado que uno de los íntimos colaboradores de este entorno realice para nosotros un análisis de sus adelantos y aportaciones.

DIV Manía ha sido acogida con un entusiasmo que agradecemos de todo corazón. Hemos recibido muchas cartas felicitándonos, así como E-mails y alguna que otra llamada telefónica. Por todo ello nos propusimos poner en la calle una revista mejor y más adaptada a las diversas peticiones que nos han llegado. El reportaje central toca ahora un tema algo más abierto que el que se centra estrictamente en DIV. Se trata de cómo hacer dinero programando videojuegos, con interesantes entrevistas a algunos de los más importantes programadores españoles del momento.

Queremos reiterar nuestra disponibilidad respecto a los lectores que quieran enviarnos sugerencias o dudas, o que quieran sumarse a la fuerte corriente de DIV que coge cada vez mayor fuerza dentro de Internet. Podéis enviar vuestras cartas a la dirección: C/Alfonso Gómez, 42, nave 1-1-2, 28037, Madrid, indicando en el sobre Correo DIV Manía, o bien enviar un E-mail al correo electrónico de la revista, divmania@prensatecnica.com.

Director: Mario Luis
mluis@prensatecnica.com
Coordinador Técnico: Rafael María Claudin
divmania@prensatecnica.com
Colaboradores: Antonio Marchal, Pablo Trinidad, José M. Sevillano, Sergio Cánovas, Miguel Barroso, David Martínez, Emilio Llanos.
Edición: Vanessa González, José Alberto Martínez
Dirección de Arte: Francisco Calero

Jefa Dpto. Maquetación: Carmen Cañas
Maquetación: Manuel J. Montes, Marga Vaquero, Silvia M. Villanueva, José A. Gil, Mª José Jiménez y Antonio Barbero
Portada: Francisco A. Anguis
Publicidad: Marisa Fernández, Sonia Glez.-Villamil, Jorge González y Noelia Menéndez
marisa@prensatecnica.com
Coord. Publicidad: Beatriz Generoso

Supervisión CD-Rom: Jesús Fernández Torres
Servicio Técnico CD-Rom: David Amaro.
Horario de atención: tardes 16:00 - 18:00 h
E-mail: stecnico@prensatecnica.com
Secretaria de Redacción: Montserrat Barreda
Departamento de Suscripciones: Sandra Fernández, Noemí Iscar
suscripciones@prensatecnica.com

Departamento de Administración: José Antonio Rivas, Juan Ignacio Domínguez y Juan López
Departamento Comercial: Marcelino Ormeño

REDACCIÓN, PUBLICIDAD Y ADMINISTRACIÓN
c/ Alfonso Gómez 42. Nave 1.1.2
Madrid 28037. España
Tfno: 91 304. 06. 22
Fax: 91 304. 17. 97
Si llama desde fuera de España, marcar (+34)

Sumario

Programas del lector

8
COMANDO PELOTA, el ganador
El vencedor de este mes es un animado juego el que tienes que conseguir que al menos una de las tres pelotas sobrevivan al incesante ataque que sufren desde las alturas.

12
FUMIGASIÓN GALÁSTICA
El segundo puesto ha recaído en un curioso y divertido programa de matamarcianos donde el humor es un elemento imprescindible del juego.

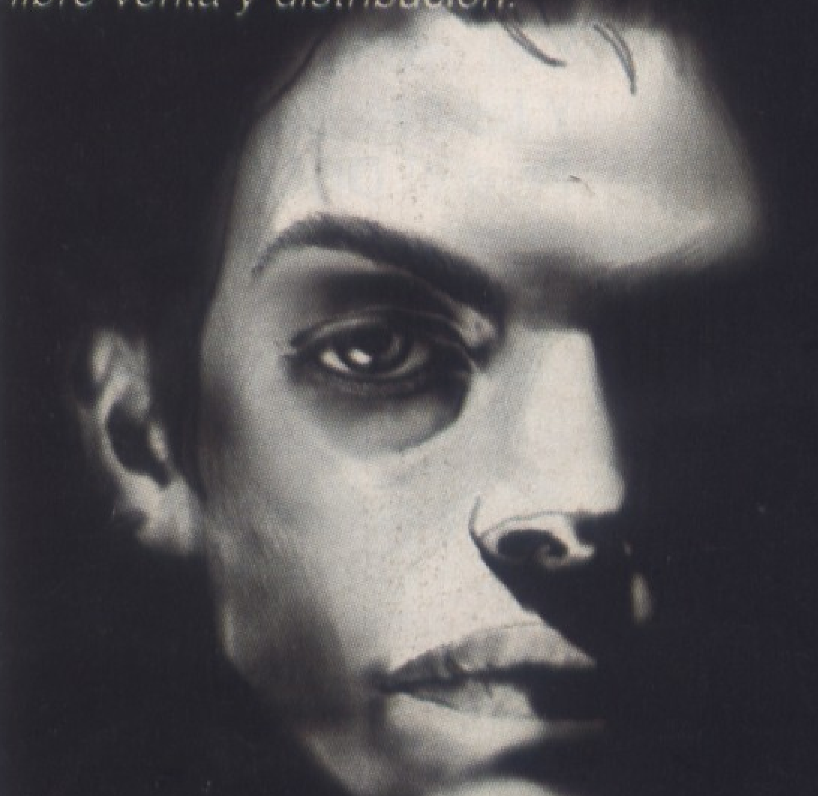
15
MANGA PARADISE, el tercero
Un puzzle se ha hecho con la tercera plaza de nuestro concurso. Entretenido y gracioso.

PREMIAMOS LOS MEJORES JUEGOS DE LOS LECTORES

DIV ha creado toda una escuela dentro del mundo de la programación. Todos los que se han acercado a la herramienta han sido capaces, de hecho, de programar. Por ello, realizamos un concurso entre los lectores que hayan realizado juegos con DIV. Os ofrecemos un primer premio de 25.000 pesetas y dos accésit de 20.000 pesetas.

¿QUÉ ES DIV GAMES STUDIO?

DIV Games Studio es una herramienta de programación que facilita en gran manera nuestra inmersión en el software de entretenimiento. Es el primer entorno profesional que permite realizar videojuegos con fines comerciales sin necesidad de un pago adicional. Con el carné de desarrollador incluido se permite el desarrollo de cualquier número de videojuegos y su libre venta y distribución.



E-mail: divmania@prensatecnica.com
http://www.prensatecnica.com
Horario de atención al público: de 09:00 a 19:00 h ininterrumpidamente

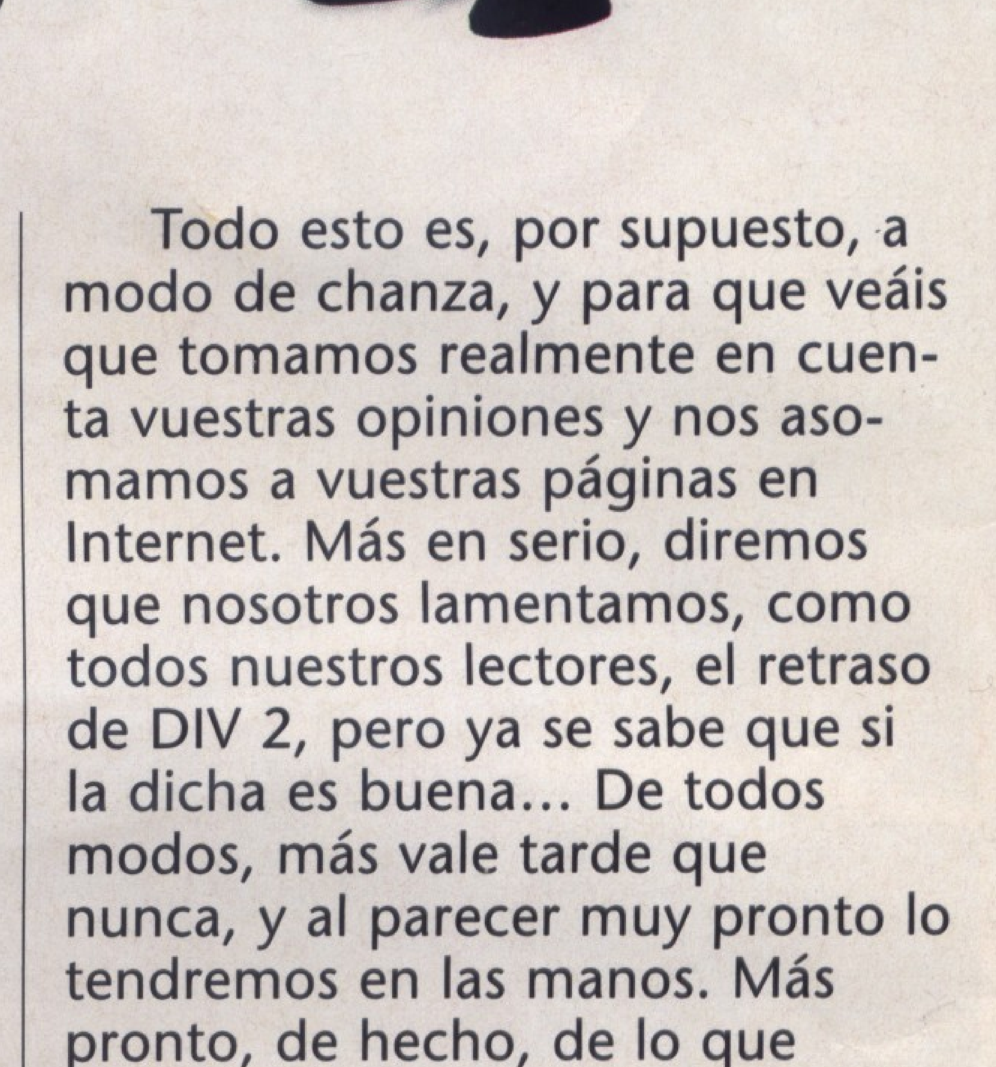
EDITA: PRENSA TÉCNICA
Director General: Mario Luis
Director Editorial: Eduardo Toribio
etoribio@prensatecnica.com
Director de Producción: Jorge Rodríguez

Directora Financiera: Felipe Hernández
Directora Publicidad: Marisa Fernández
Director Comercial: Esteban Martínez
Fotomecánica: FCM
Impresión: Printerman
Duplicación del CD-Rom: M.P. O., Servicios Ibéricos, Grupo Cóndor

Distribución:
SGEL. Avda Valdelaparra, 29
Alcobendas. Madrid

Depósito legal: M-42077-1998
AÑO 1 • NÚMERO 2

El lector lúdico



Sólo para aficionados a los juegos

En fiel reflejo de nuestros lectores y de todos aquellos que estén relacionados con DIV nos queremos convertir, y para ello ponemos a vuestra disposición esta página de la revista, además de otras en las que esperamos vuestra participación.

Así pues, en primer lugar queremos salir al paso del pequeño revuelo que se ha organizado acerca de la tardanza de DIV 2. Por ello, hemos elegido uno de los más jocosos alegatos que hemos encontrado dentro de la Red de redes, que ofrece cobijo a muchas de las más puras manifestaciones en torno a DIV. El poema aparece en la web de Deemo, está escrito por Fidojones y describe con gracia la tardanza de DIV. Los versos dicen así:

Y llegaron las Navidades,
Y los Reyes Magos,
Y el orondo bonachón,
Y las Pascuas pasaron,
Pero el DIV 2 no llegó.

Y llegaron las lluvias,
Y llegaron los nubarrones,
Y vinieron los vientos del Norte,
Y los copos de nieve,
Pero el DIV 2 no llegó.

Y comenzaron los carnavales,
Y llegó don Carnal,
Y la gente loca en las fiestas,
Y comenzó la Cuaresma,
Pero el DIV 2 no llegó.

Y vendrá la Semana Santa,
Y llegará la penitencia
Y vendrá Abril con sus aguas mil,
Y saldrán los santos en procesión,
Pero el DIV 2 no llegó.

Nosotros, con toda nuestra humildad, hemos contestado a este original Fidojones, con otros versos, que defienden, desde luego, a DIV 2, y que no pretenden otra cosa que responder de la misma forma jocosa y farandulera a un divero de pro que, estamos seguros, espera regocijado la, ahora sí, inminente aparición de DIV 2. Nuestros versos dicen así:

Era un señor, y como señor
Dejó que pasara el tiempo,
Y con griterío, y con dolor,
rompimos nuestro silencio.

Pues sentíamos el compilador,
Como, en lo más negro, un beso,
Y queríamos probar el sabor
De las librerías sin freno.

Y nuestro compartido temor
Se expresó en dos poemas negros,
que del gran DIV decían, sin pudor,
que era, a la izquierda, sólo un cero.

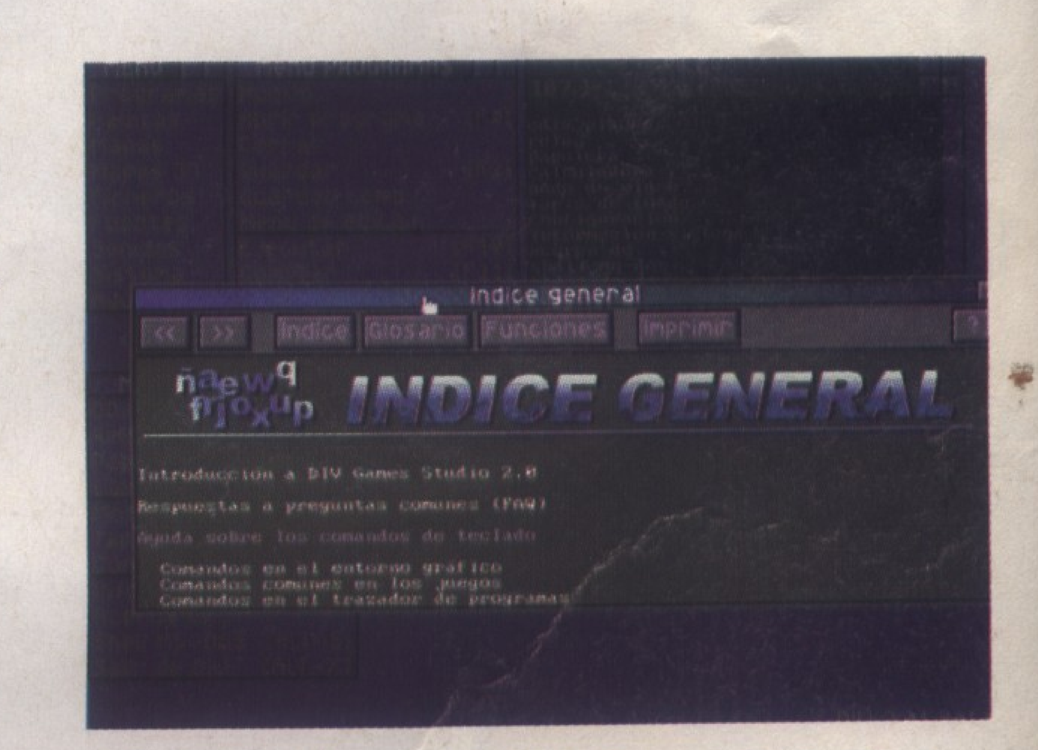
Iba llegando ya el calor,
Apareció el DIV ameno.
Era, en la serie, el número dos,
un programa muy certero.

Se acalló ya, por fin, el furor
De los muy fuertes lamentos,
Había llegado nuestro señor
Sobre sólidos cimientos.

Todo esto es, por supuesto, a modo de chanza, y para que veáis que tomamos realmente en cuenta vuestras opiniones y nos asomamos a vuestras páginas en Internet. Más en serio, diremos que nosotros lamentamos, como todos nuestros lectores, el retraso de DIV 2, pero ya se sabe que si la dicha es buena... De todos modos, más vale tarde que nunca, y al parecer muy pronto lo tendremos en las manos. Más pronto, de hecho, de lo que muchos se imaginan.

A los usuarios impacientes les remitimos a las páginas de la revista que tratan sobre DIV 2. Están escritas por alguien que ya se ha movido en este entorno, y lo conoce a fondo; de modo que podéis esperar lo mejor de ellas. Se trata, desde luego, de Tizo, un colaborador que todos vosotros, si navegáis de cuando en cuando, conoceréis de sobra. Por lo demás, sólo podemos señalar que en el momento en que la revista esté en la calle, ya tendréis DIV 2 y, si no es así, estará a punto de salir.

Esperamos recibir en el futuro todas vuestras críticas y vuestras sugerencias, y si son tan constructivas como la de Fidojones, entonces mejor que mejor. Un saludo a todos.



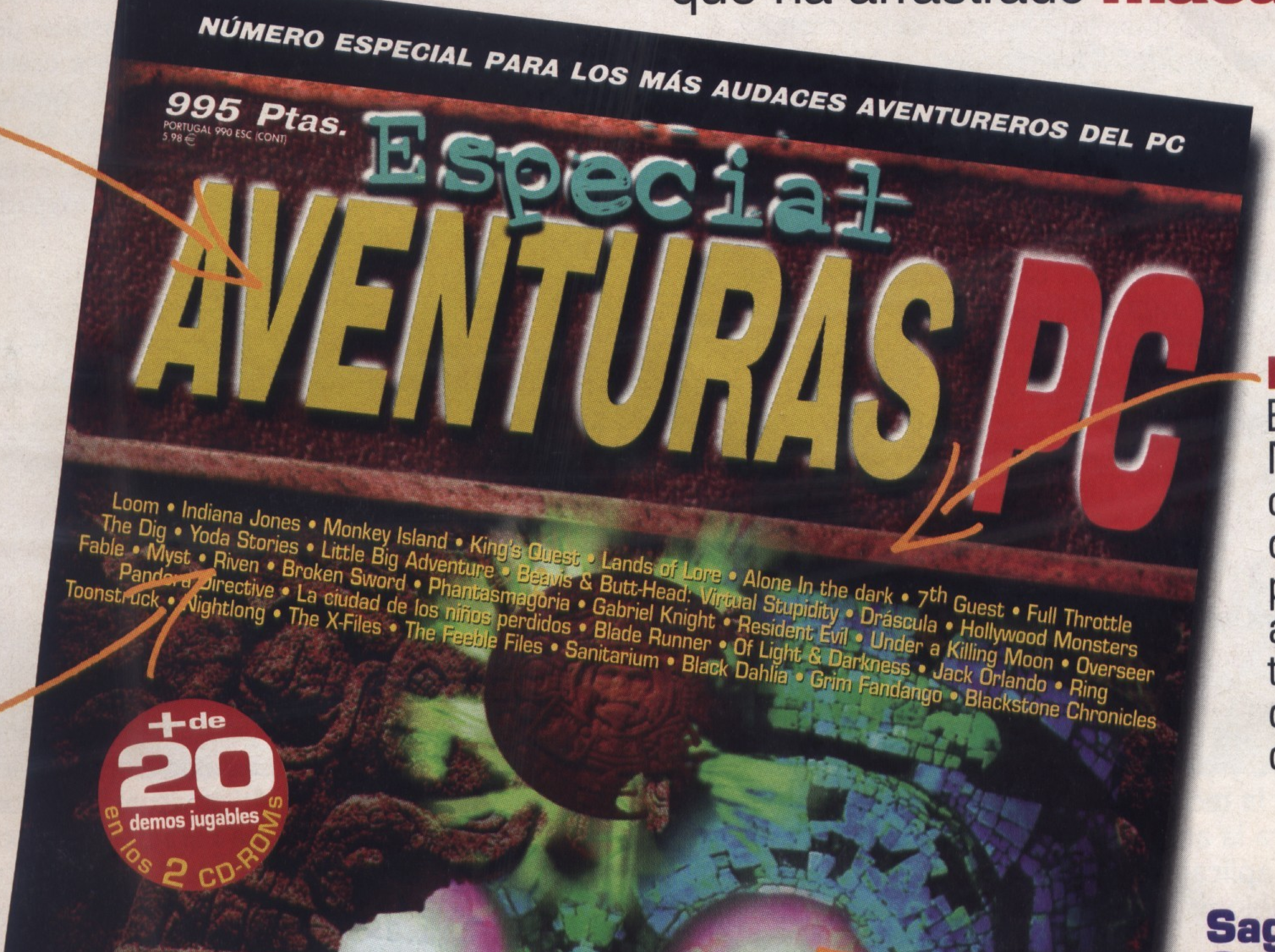
¡Tranquilizaos! DIV 2 pronto estará en vuestros cocos. La espera bien ha valido la pena.

(*) DIVadicto, DIVo, DIVa: Dícese de aquel aficionado a la programación de videojuegos que se encuentra fascinado por DIV. ImpeDIVmentos: En lenguaje de DIVos, se trata de los problemas a los que se enfrenta un desarrollador que use DIV.

Para **revivir** grandes **hazañas,** aventúrate en un **especial** de **fábula**

Las **últimas apariciones** en el mercado **lúdico**

Comentarios **detallados** de los **hitos** de un **género** que ha arrastrado **masas**



eportaje
iaje a ninguna arte. Las mejores ideoaventuras de odos los tiempos.

urse of Monkey sland
Última parte de las venturas de Guybrush: LeChuck, ncluso desde el eino de los nuertos, hace de as suyas a uno de os románticos héroes jamás creados.

Sanitarium
Erbe nos ha traído una de las mejores aventuras, con un look siniestro y terrible.

Ring
El anillo de los Nibelungos sirve como algo más que inspiración para una aventura. Un título que no debemos perder de vista.

Saga King ´s Quest
Una de las más importantes series de Sierra y de Roberta Williams, polémica desde su mismo nacimiento, que sigue manteniendo viva la llama que la ha hecho famosa.

Riven
La continuación del genial Myst, uno de los más originales programas de género.

Comentamos entre otros:
• Curse of Monkey Island • Saga King ´s Quest • Resident Evil 1 y 2 • Overseer • Blade Runner • The X Files • The Feeble Files • Grim Fandango • Ring • Riven

Especial AVENTURAS PC — CD1

DEMOS
- Broken Sword
- Broken Sword 2
- Alone in the Dark
- Muppets Treasure Island
- Monty Python & the Holy Grail
- Discworld 2
- Jack Orlando
- Last Express
- 3 Skulls of Toltecs
- Ecstatica 2
- Zork Grand Inquisitor
- Indiana Jones y la última cruzada
- Beavis & Butt-Head: Virtual Stupidity

Video
- Riven

Especial AVENTURAS PC — CD2

2 CD-Roms con algunos de los **mejores títulos** de la **historia** de este género; incluyen **más** de **20 demos jugables** y **vídeos** de programas como **Grim Fandango, Curse of Monkey Island, Phantasmagoria I y II, Gabriel Knight 2, Broken Sword I y II, etc.**

Prens@ Técnic PRENSA TÉCNICA • C/ Alfonso Gómez, 42 - Nave 1-1-2 • 28037 Madrid España (Spain) • Tfno : 91. 304.06.22• Fax : 91. 304.17.97
www.prensatecnica.com

Concurso Made with Macromedia

Macromedia presenta su prestigioso galardón Made with Macromedia (Premio del Público) que se celebrará paralelamente a la tercera edición de Europe UCON'99. Este año tendrá lugar en Disneyland París los días 24 y 25 de junio de 1.999.

El concurso está abierto a los proyectos creados con una o varias herramientas Macromedia (Authorware, Director, Dreamweaver, Fireworks, Flash, Generator, Freehand, y Fontographer) y significa el escaparate perfecto para mostrar el talento de los desarrolladores y diseñadores españoles de multimedia, gráficos y Webs en toda Europa.

Las ocho categorías del concurso son: presentación de negocio, proyecto multimedia educacional, proyecto de formación empresarial, proyecto de consumo, proyecto de entretenimiento, "Kidsware" específicamente destinado a los niños, proyectos artísticos y de marketing. También habrá premio para el mejor proyecto llevado a cabo por estudiantes y para el mejor sitio Web, los mejores gráficos e ilustraciones y el mejor diseño comercial.

El proyecto no puede haberse iniciado antes de junio de 1998 ni haber participado en ediciones anteriores, siendo el plazo de presentación hasta el día 16 de abril de 1999. Se pueden presentar online o en CD-ROM. Los trabajos de los tres finalistas en cada una de las ocho categorías se mostrarán en la Web de Macromedia, una de las más visitadas de la red. Sin duda serán el centro de atención de cientos de diseñadores, desarrolladores y medios de comunicación.

Los interesados pueden obtener más información llamando al teléfono 93-237 12 08. También se puede enviar un E-mail a EUCON@macromedia.com o visitar la Web *www.macromedia.com/macromedia/international/events/*

Integración para la tecnología 3DNow!™ de AMD

Los más importante desarrolladores y editores de hardware y software han anunciado su intención de integrar la tecnología 3DNow!, la primera innovación para la arquitectura x86 que mejora de forma significativa las aplicaciones de gráficos 3D y multimedia. Esta nueva tecnología hace uso intensivo de la coma flotante, utiliza SIMD(datos múltiples de instrucción simple), así como otras mejoras de rendimiento con el fin de permitir una experiencia informática visual superior.



Introducida como una característica principal del procesador AMD-K6-2 en mayo de 1998, la tecnología 3DNow! lleva una ventaja de más de nueve meses en cuanto a su salida al mercado sobre sus competidores tecnológicos de mejora de gráficos 3D basados en CPU.

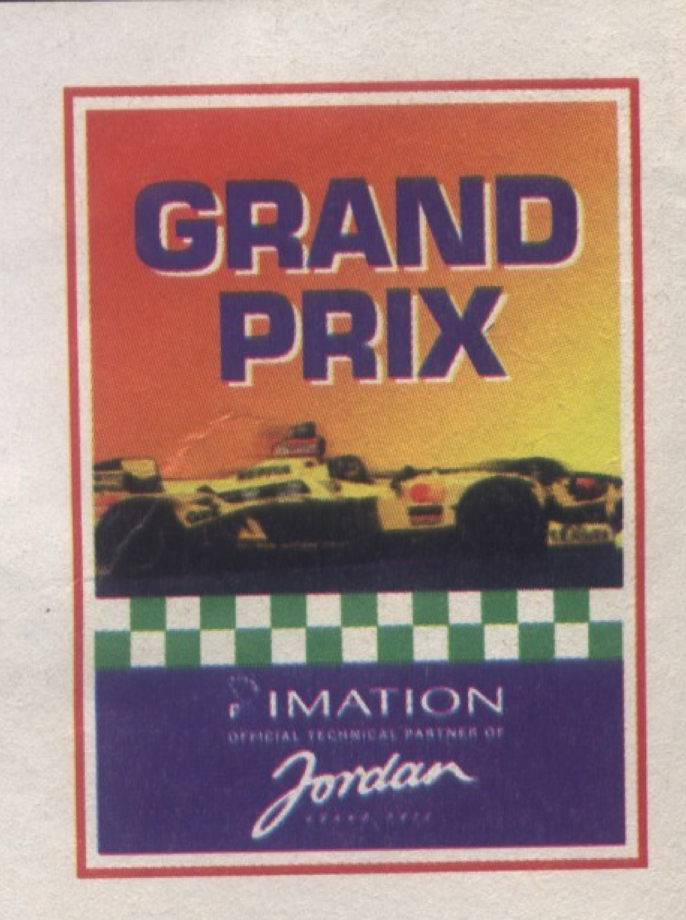
Se han optimizado ya numerosos productos de hardware y software para la tecnología 3DNow!. Accolade, Activision, Criterion Studios, Digital Anvil, Eidos, GT Interactive, Gremlin, id Software, Interplay, Psynosis, Rage y 3DO han anunciado recientemente su intención de ofrecer soporte para esta tecnología en varios de sus futuros títulos de software. Dichos editores y desarrolladores se han unido a una lista en constante crecimiento de desarrolladores de software y hardware, incluidos Microsoft, IBM, Epic Games, 3Dfx, y Matrox, que ya soportan la tecnología 3DNow! en sus principales aplicaciones 3D, las cuales hacen un uso intensivo de gráficos, y en su hardware.

Día a día se siguen añadiendo nuevos títulos que soporten esta tecnología. Para obtener una lista completa de los productos que la soportan consulte la página Web de AMD: *www.amd.com/products/cpg/k623d/optimized.html*

CeBIT 1999

Imation exhibió en Cebit 99 sus últimos avances tecnológicos y productos para la gestión de información y almacenamiento de datos, incluidas las últimas novedades de su gama de productos para PCs, redes y centros de proceso de datos. Éstos se expusieron en un satand dinámico diseñado basándose en la reciente asociación técnica entre Imation y el equipo Jordan Grand Prix de Fórmula Uno.

Imation presentó en esta edición de la feria, celebrada el mes pasado en Hannover, unas cintas específicamente diseñadas para el entorno empresarial y de redes de gama alta: Royal Guard 9490EE.



El cómic de Lara sólo para Francia

Sólo los franceses podrán disfrutar de las aventuras en papel de Lara. Según informa Game Daily, Lara Croft protagonizará una serie de cómics franceses. Los conocidos dibujantes Alex Alice y Patrick Fréon serán los responsables de trasladar las curvas virtuales de Lara al papel. De momento no habrá traducción a otros idiomas, aunque todo dependerá del éxito del cómic para que esto suceda.

El guión de los cómics seguirá la tónica de las aventuras en los videojuegos de Lara Croft, muy al estilo Indiana Jones. Aunque se espera mucha acción, los cómics serán una buena forma para otorgarle un pasado y unos sentimientos a la heroína, ya que en los videojuegos no dice mucho de sí misma, más bien se limita a una fachada de chica atractiva.

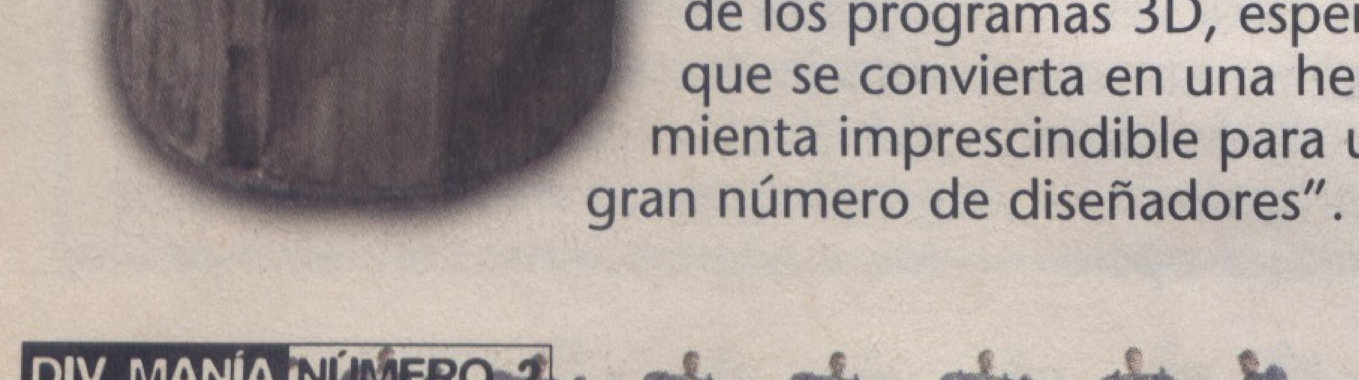




MetaCreations anuncia Bryce 4

La nueva versión de Bryce, uno de los mejores software para la creación de mundos y paisajes 3D para PC y Mac, ya está en el mercado. Bryce 4 continúa expandiendo las opciones creativas para profesionales gráficos, diseñadores de Web y programadores de juegos. Ahora, incluye una mayor compatibilidad con otro software de programación en 3D, además de nuevas opciones.

Bryce se ha convertido en una herramienta indispensable para los profesionales del 3D. Esta cuarta versión viene a mejorar las anteriores en diferentes aspectos. La posibilidad de exportar texturas desde aplicaciones de animación como LightWave, Ray Dream Studio, e Infini-D; la facilidad para crear objetos, texturas, modelos y escenas; y la gran ventaja de resultar intuitivo para los no iniciados en el mundo 3D. Los profesionales, artistas y animadores podrán mejorar su habilidad para crear escenas realistas y luego animarlas. Según John Leddy, vicepresidente de MetaCreations, “Bryce ha sido durante mucho tiempo el preferido de muchos diseñadores y animadores en 3D. Ahora que es compatible con muchos de los programas 3D, esperamos que se convierta en una herramienta imprescindible para un gran número de diseñadores”.



Exportar escenas de HTML con imágenes mapeadas ahora será más fácil. Entre las nuevas opciones se incluye la posibilidad de bajar películas en RealPlayer, VRML y QTVR desde la Web. Todo lo nuevo de Bryce puedes encontrarlo en su Web, que provee a los usuarios instantáneamente de información e incluso tiene un forum para los visitantes llama Bryce Talk Area. Este área incluye soporte técnico personal para clientes y la posibilidad de insertar nuestros links.

Nuevas y mejoradas opciones para importar objetos. Efectos de sombra de las nubes. Compatibilidad con todos los modelos de animación y modelación ambientadas en 3D. Soporte para exportar todo tipo de áreas texturizadas. Estas nuevas opciones vienen a sumarse a la nueva versión de Bryce. Quizá, lo más novedoso sea la posibilidad de acceso al Instituto Geológico de Medición de los EE.UU. en Internet, con facilidad para importar a Bryce las áreas y texturas de los mapas reales.

Los usuarios pueden encontrar, dentro del CD distribuido con el programa una increíble colección de escenas, animaciones, objetos, tutoriales y más.



DIV 2: Versión definitiva

La nueva herramienta

Aunque ya se analizaron en otro artículo las excelencias y desgracias de DIV 2, debido a cambios de última hora, no se comentó la herramienta definitiva. Este mes haremos un análisis con la versión beta en la mano. Recorreremos todas las funciones nuevas del lenguaje y revisaremos el entorno para observar las nuevas mejoras.

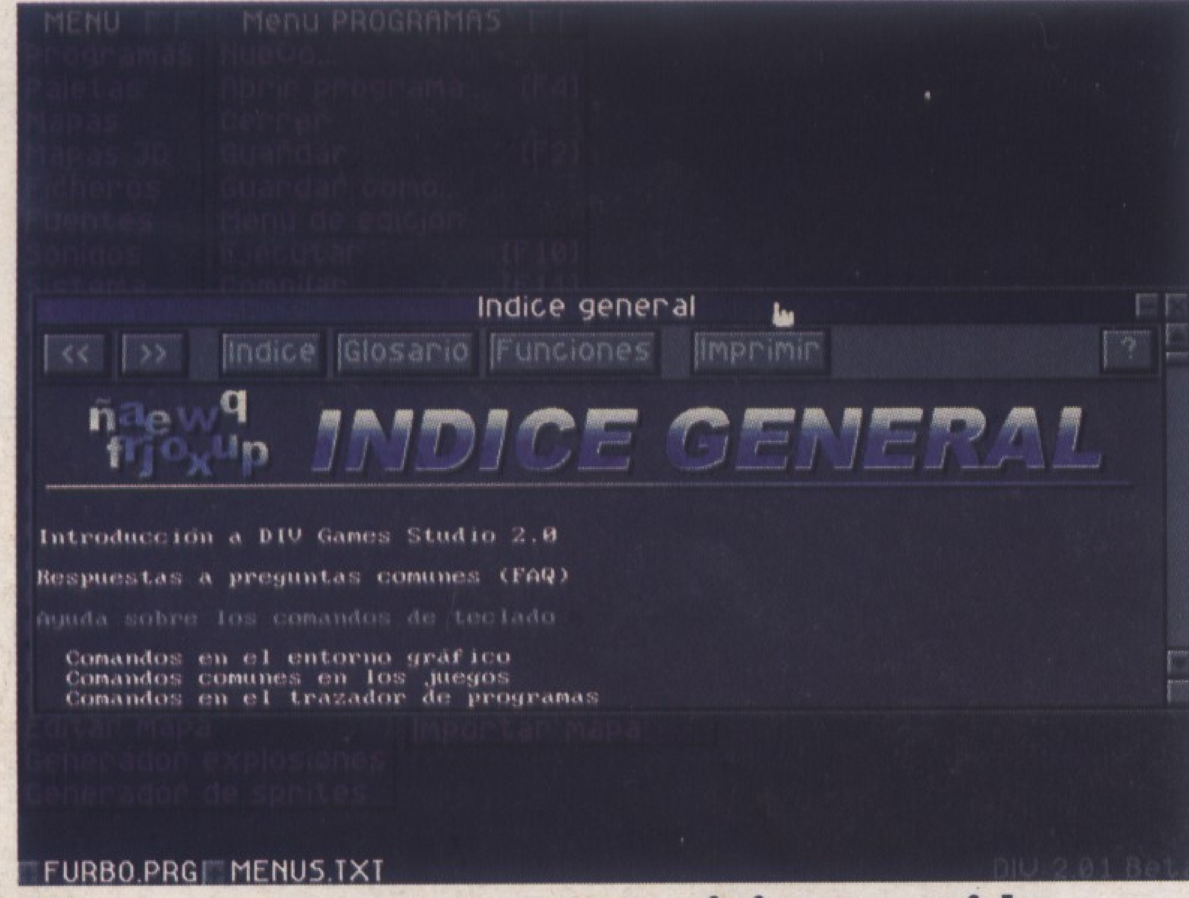


DIV 2: Un look para el siglo XXI.

Este artículo intentará ser lo más objetivo posible. Aunque, para qué negarlo, nos ligan ciertas relaciones con la herramienta y siempre se debe tener una opinión crítica sobre cualquier tema.

> La opción de mapas 3D es la más novedosa dentro del menú de inicio del programa

Como se ha dicho, en el anterior artículo no se tenía la herramienta definitiva; en éste, se recorrerán una a una todas las novedades. Para que todo resulte más sencillo, y no perdernos entre tanta mejora, iremos recorriendo de una manera especial la herramienta. En primer lugar, visitaremos todos los menús de DIV para ver las mejoras incluidas. Y, si existe alguna otra mejora que esté relacionada con ese menú, pero que no sea visible, también se comentará. Por lo tanto, veremos las mejoras del entorno, comentando también el editor gráfico. Luego recorreremos toda la lista de funciones para ver las novedades, esta vez en el lenguaje. Usando este método no nos dejaremos nada en el camino, además ver sólo las mejoras realizadas, ni una más, ni una menos. Pero empecemos sin mas preámbulos a ver todas las novedades.



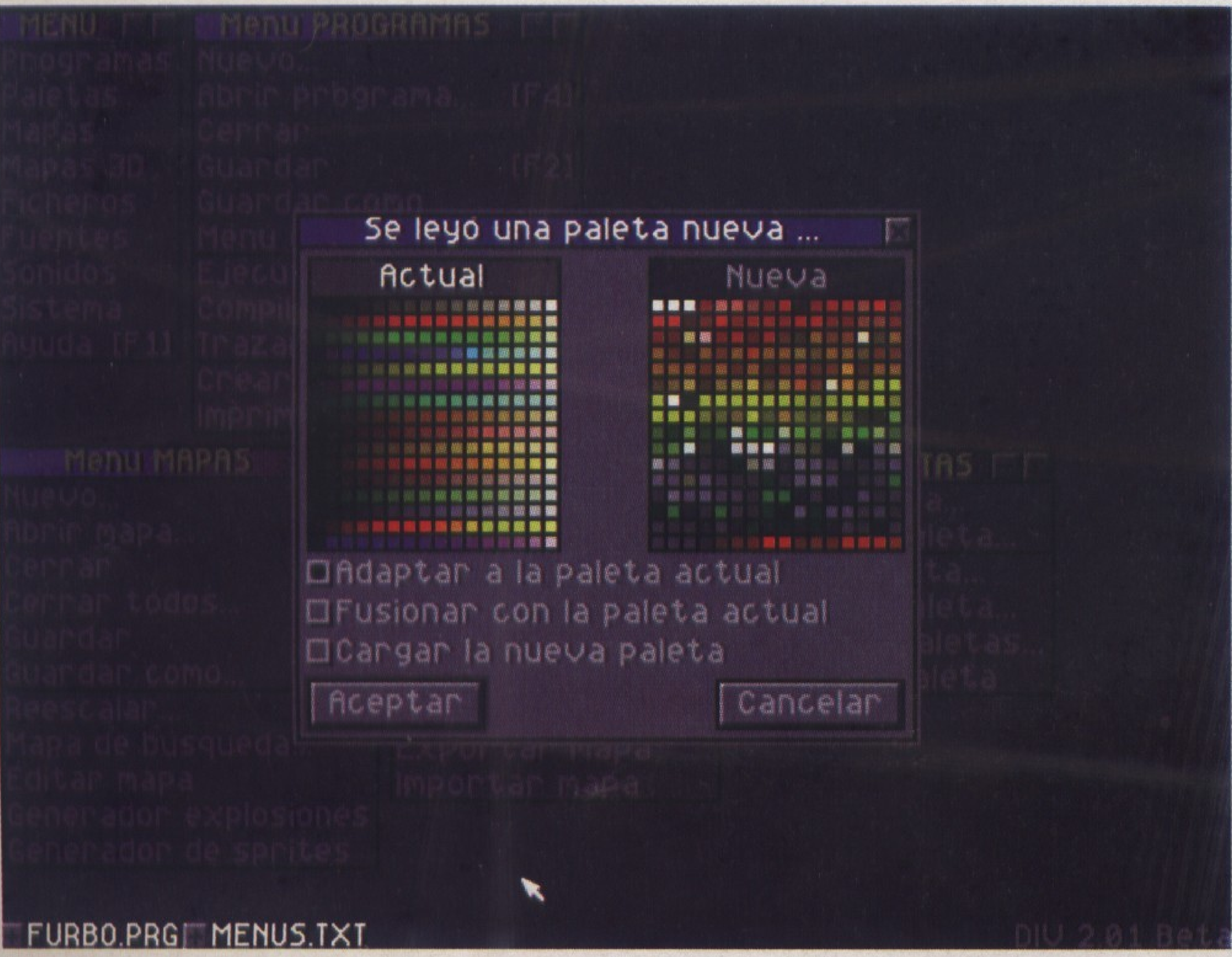
Un entorno que es ya un viejo conocido, ahora con grandes mejoras.

El entorno

Como ya se ha dicho, se verá cada uno de los menús disponibles, empezando por el principal. Las opciones de las que consta son: *programas, paletas, mapas, mapas 3D, ficheros, fuentes, sonidos, sistema y ayuda.* La única reseña novedosa que se puede ver es la opción de *mapas 3D*, que veremos más adelante. Los otros menús son ya viejos conocidos, aun así, los visitaremos para ver las novedades.

El primero de ellos, siguiendo el orden, es el de menú *programas*, que consta de las siguientes opciones: *nuevo, abrir programa, cerrar, guardar, guardar como, menú de edición, ejecutar, compilar, trazar programa, crear instalación e imprimir listado.* Lo primero, decir que ha cambiado la forma de manejar los ficheros, pero esta novedad se verá más detenidamente en otros menús. El *trazador de programas* también ha tenido ligeros cambios, mostrando ahora mucha más información. Además, existe la nueva opción de poder imprimir el listado. Otra pequeña modificación que se ha hecho en el editor es que ahora incluye la posibilidad de colorear los listados. También la función *Ir a...* es más sensitiva, pues si nos ponemos encima de un proceso lo busca en la lista de funciones. No hay que olvidar la *instalación de programas*, de la cual se ha hecho una renovación bastante completa. Permite elegir entre ficheros empaquetados o sueltos, código optimizado para Pentium, etc. Además permite dar el *look* que queramos a nuestras instalaciones mediante el cambio de algunos ficheros gráficos. Todo ello da a las instalaciones un aspecto más profesional.

El siguiente menú es el de *paletas*, cuyas opciones son: *abrir paleta,*



Incluso han mejorado el menú de cambio de paleta.

guardar paleta, editar paleta, ordenar paleta, fusionar paletas y mostrar paleta. El menú en sí, no ha cambiado, pero sí la forma de tratar las paletas. Lo primero es que el *browser* nos enseña las paletas que vamos a cargar. Además, a la hora de trabajar con mapas o ficheros, se podrán cargar todos ellos y crear una paleta unificada. También decir que se han optimizado los procesos de *adaptación* y *fusión* de paletas.

El siguiente menú es el de *mapas*, que sí tiene grandes cambios. Las opciones que nos encontramos son: *nuevo, abrir mapa, cerrar, cerrar todos, guardar, guardar como, reescalar, mapa de búsqueda, editar mapa, generador explosiones y generador de sprites.* Pero vayamos por partes, hablando primero de lo que no se ve en dicho menú. El primer detalle es el *browser* de mapas incluido en la ventana de manejo de ficheros. Podremos ver todos los mapas en forma de *thumbails*, es decir, una imagen

DIV 2: Lo que entra por los ojos

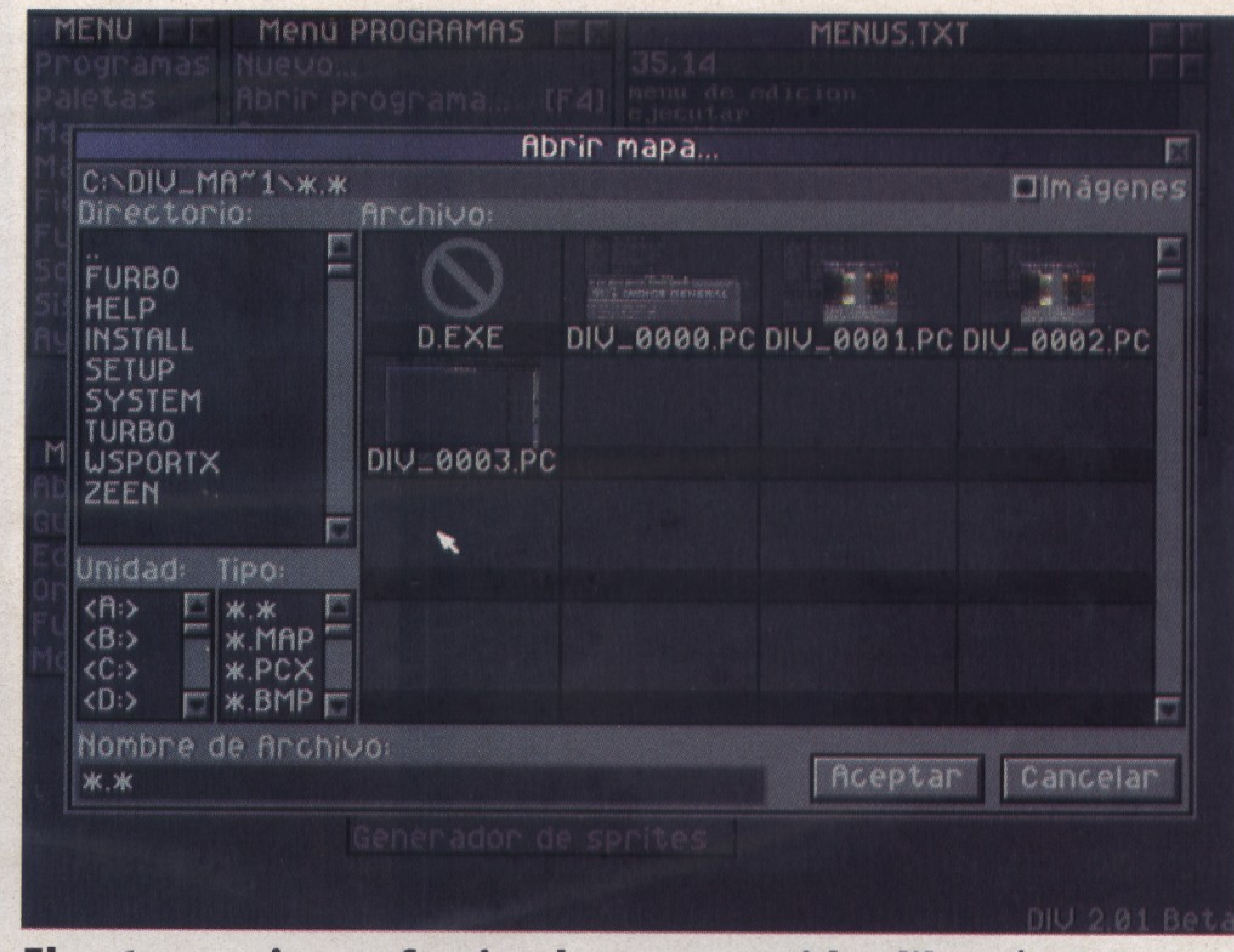
Siempre que se prueba alguna herramienta nueva, hay algunos aspectos que llaman más la atención del usuario que otros. Veamos algunos de los nuevos aspectos de DIV 2 que cumplen este cometido:

- *Listado coloreado*: Ahora todo será mas legible ya que los listados se verán de colores, indicando las distintas partes del listado: comentarios, palabras reservadas, símbolos....
- *Cargador con Thumbails*: Cuando se cargue un mapa o una paleta se podrá visualizar antes desde el cargador. Además, se ha variado la visualización de ficheros FPG.
- *Mundos 3D*: Algo que llama mucho la atención son los *modos 8*. Y es que, dentro de un mundo tipo Doom, se puede meter casi cualquier juego.
- *Las instalaciones*: Este aspecto ha cambiado muchísimo tanto visualmente como funcionalmente. Es totalmente configurable, además de proteger mejor nuestro trabajo.
- *Funciones de IA*: Para juegos de estrategia. Las funciones de búsqueda de caminos resolverán un gran problema.
- *Funciones de textos y ficheros*: Algo que se echaba en falta en la antigua versión y que en ésta ha sido arreglado con creces.
- *El sonido*: Ha sido reconstruido totalmente permitiendo una mayor calidad. Además, ahora podemos poner canciones a nuestro juegos.

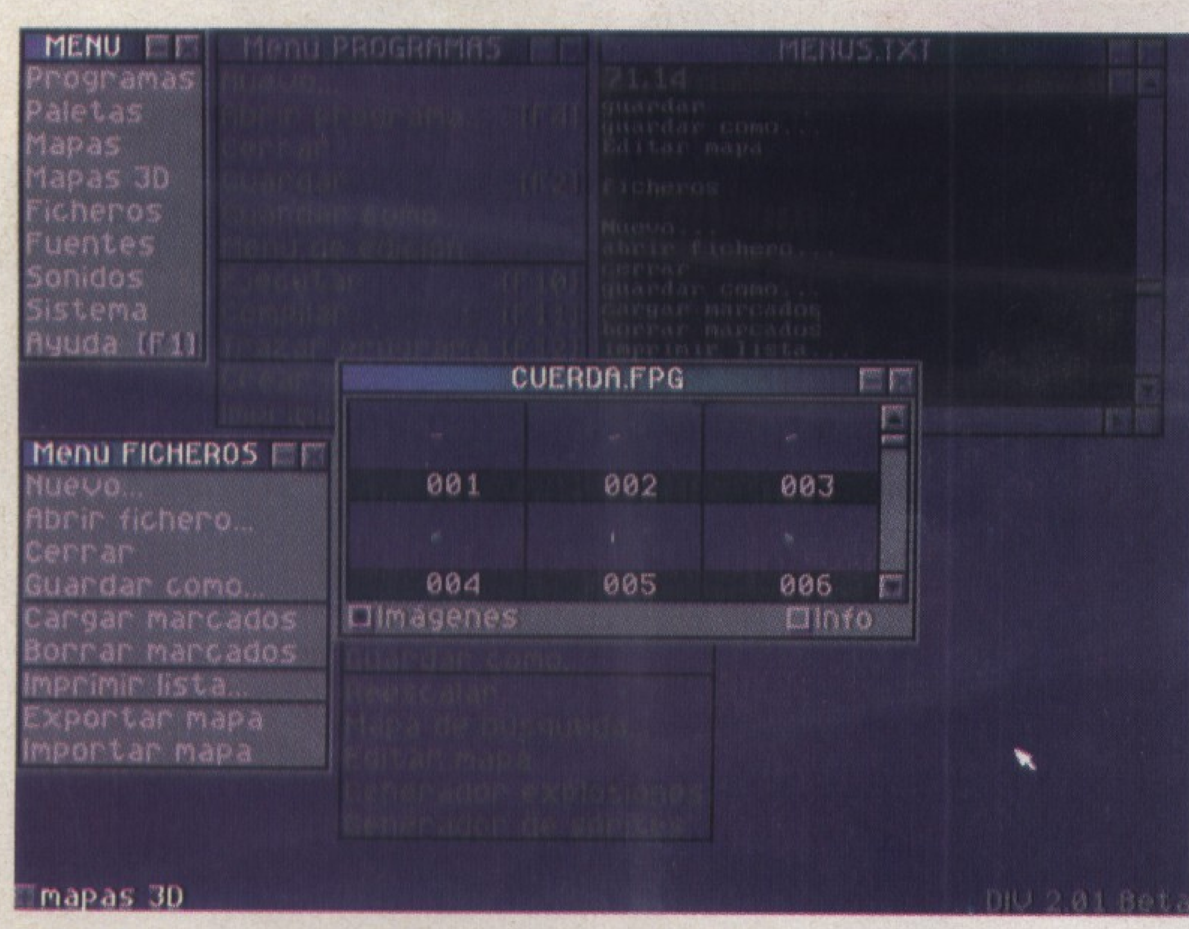




La opción más esperada: editando un mapa 3D.



El entorno sigue ofreciendo unas surtidas librerías.

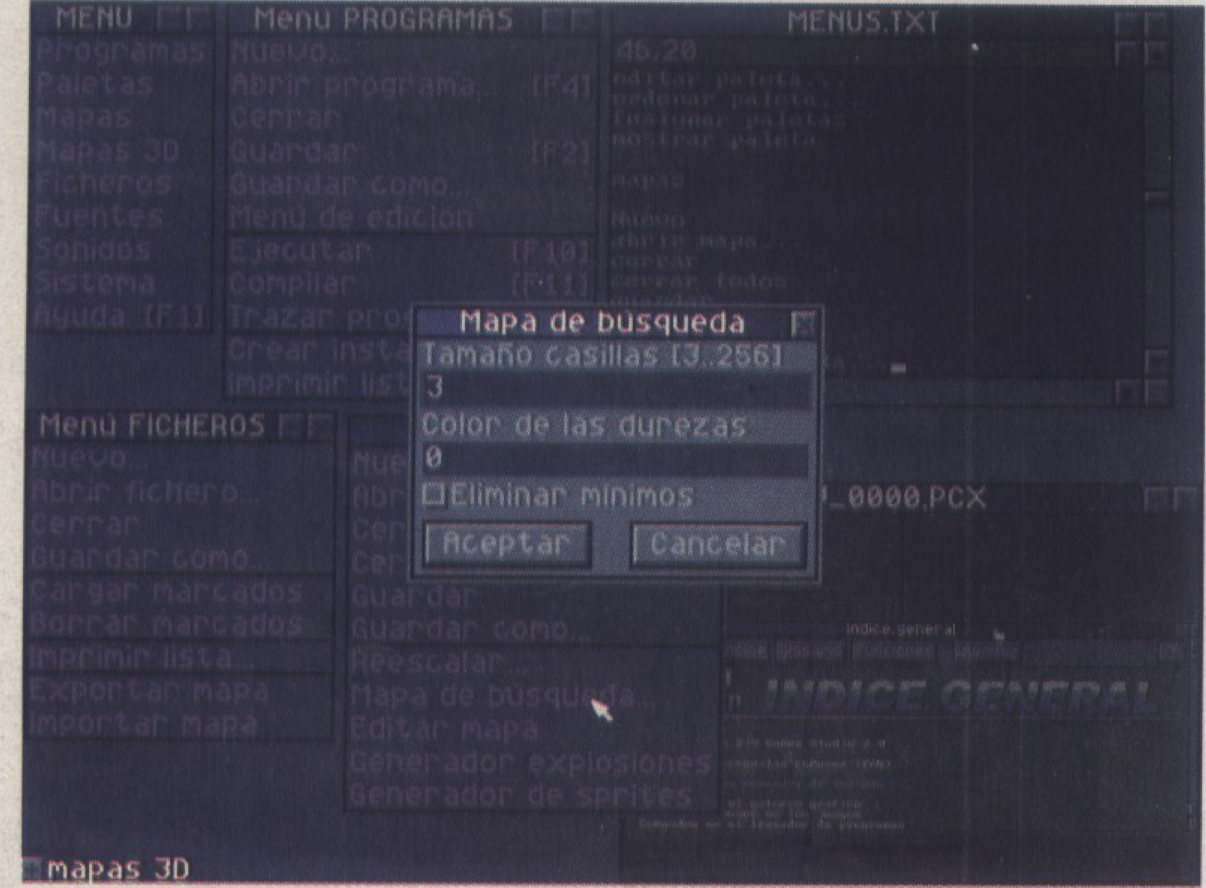


Distintos modelos para objetos comunes.

reducida de los mismos. Con este sistema ya no se nos perderán los gráficos. Además, se permite todo tipo de *tageos*, pudiendo seleccionar y cargar varios ficheros a la vez. Ahora hablemos de las opciones del menú que tienen alguna novedad. La primera de ellas es la de los *mapas de búsqueda*. Estos mapas son necesarios para las rutinas de inteligencia artificial, para búsqueda de caminos. Con dicha opción se generarán estos mapas de búsqueda.

El editor de sonidos permite cortar, pegar, reproducir y algunas opciones de efectos

Antes de pasar a otras opciones, comentar que el *editor de mapas* también ha tenido alguna mejora.



Es importante corregir los errores.

Subrayar el hecho de que podemos usar cualquier tipo de pincel para pintar. Además, se podrán tener dos capas, una de ellas de fondo, para poder realizar mejor nuestro trabajo artístico. Y hablando de trabajo artístico, la otra innovación en el menú *mapas* es el *generador de sprites*. Esta herramienta consiste en una especie de cámara que permite capturar animaciones. Disponemos de tres modelos, hombre, mujer o niño, a los cuales se les puede aplicar una textura y por último, únicamente se debe elegir la animación y el ángulo de captura de la misma. La herramienta nos generará la animación completa, muy útil para los poco hábiles con el pincel virtual.

El siguiente menú es de los más novedosos, ya que no existía en la antigua versión. Se trata de los *mapas 3D*. Las opciones que nos encontramos, habituales en otros menús son: *nuevo, abrir mapa 3D, cerrar, cerrar todos, guardar, guardar como y editar mapa*. Cuando creamos un nuevo mapa, se abre una ventana con una visión del mismo. Se puede entrar en un editor que permite crear y editar vértices, paredes y habitaciones, con el cual también podremos elegir las texturas a visualizar. Habrá también unos parámetros de suelo y techo para poder crear escalones, doseles, ascensores o puertas.

El menú *ficheros*, como no, cuenta con novedades. Las opciones son parecidas a las que había en la antigua versión: *nuevo, abrir fichero, cerrar, guardar como, cargar, marcados, borrar marcados, imprimir lista, exportar mapa e importar mapa*. Antes de nada decir que la forma de visualizar los ficheros FPG ha cambiado. Ahora existe la posibilidad de poder ver *thumbails* o reducciones de los gráficos incluidos en el fichero. Además, como hemos visto, existen dos nuevas opciones en el menú que están relacionadas. Esto permite pasar ficheros FPG a mapa gráfico, y a la inversa. La forma de hacerlo es usando gráficos limitados por una caja para indicar el tamaño. Una opción muy útil, como sabrán muchos de los que han trabajado con DIV.

El menú *fuentes* no tiene ningún cambio. Su contenido es similar a la versión anterior: *generar fuente, abrir fuente, cerrar, escribir texto, exportar mapa e importar mapa*. La única novedad es que se ha incluido una herramienta externa a DIV, que permite pasar fuentes *True Type*, tan usadas con Windows, a formato IFS, que es el que permite leer el generador de fuentes de DIV.

Para todos aquellos con vocación musical, el siguiente menú les traerá grandes sorpresas ya que es el de *sonidos*. Las opciones disponibles son: *nuevo, abrir sonido, cerrar, guardar, guardar como, editar sonido, ajustar volumen, abrir canción y cerrar canción*. La primera novedad viene con el *editor de sonidos* que permite cortar, pegar, reproducir, y otras opciones de efectos. Decir, que el sistema de sonido de DIV, ha cambiado, pudiéndose usar WAV, a calidad 44.000, estéreo y 16 bits. También se han incluido otros formatos, más musicales, como MOD, S3M y XM.

El siguiente menú, y último que dispone de opciones, es el de *sistema*. Algunas opciones son viejas conocidas, aunque todas ellas tienen alguna mejora: *cdiv player, reloj, papelera, calculadora, modo de vídeo, tapiz de fondo, configuración, información sistema, acerca de, shell ms-dos y salir de div*. Lo primero reseñable es la inclusión de una calculadora. Además se ha mejorado la compatibilidad *VESA* en toda la herramienta. Se ha cambiado el tapiz de fondo y el menú de configuración con opciones como el coloreado de textos.

Hazte Programador

DIV

Queridos lectores y aficionados a la programación: desde ahora podéis entrar a formar parte del Club DIV, para lo que tenéis que enviarnos el cupón adjunto, escribiendo "Club DIV" en el sobre, a la calle Alfonso Gómez, 42, Nave 1-1-2, 28037 Madrid. Éstos son los ganadores de 10 suscripciones a DIV Manía y 10 Snow Wave, en un sorteo realizado entre los miembros del Club DIV:

Suscripciones:
Susana Lara Cañaberas (Madrid)
Jacob Almagro García (Cádiz)
Alejandro Martín Parra (Málaga)
Luis Carlos Campos Martínez (Valencia)
Eduardo Verdugo Cáceres (Madrid)
Pedro González Bravo (Granada)
Christian López Garra (Sevilla)
Gabriela Sánchez Blasco (Madrid)
Pablo Campos Villa (Málaga)
Ricardo Amigo Blanco (Jaén)

Snow Wave Avalanche:
José Vargas Martínez (Barcelona)
David Garcías Bosch (Baleares)
Jaime Piedra Rivas (Oviedo)
Diego Darriba López (Lugo)
Ferrán Tarrdellas Llaudó (Barcelona)
Santiago Ameno Sainz (Madrid)
Carla Velasco López (Valencia)
Alberto Riego Corazón (Granada)
Alfonso Cavada Gómez (Sevilla)
Ricardo Lomas Grande (Madrid)

Carné de Programador



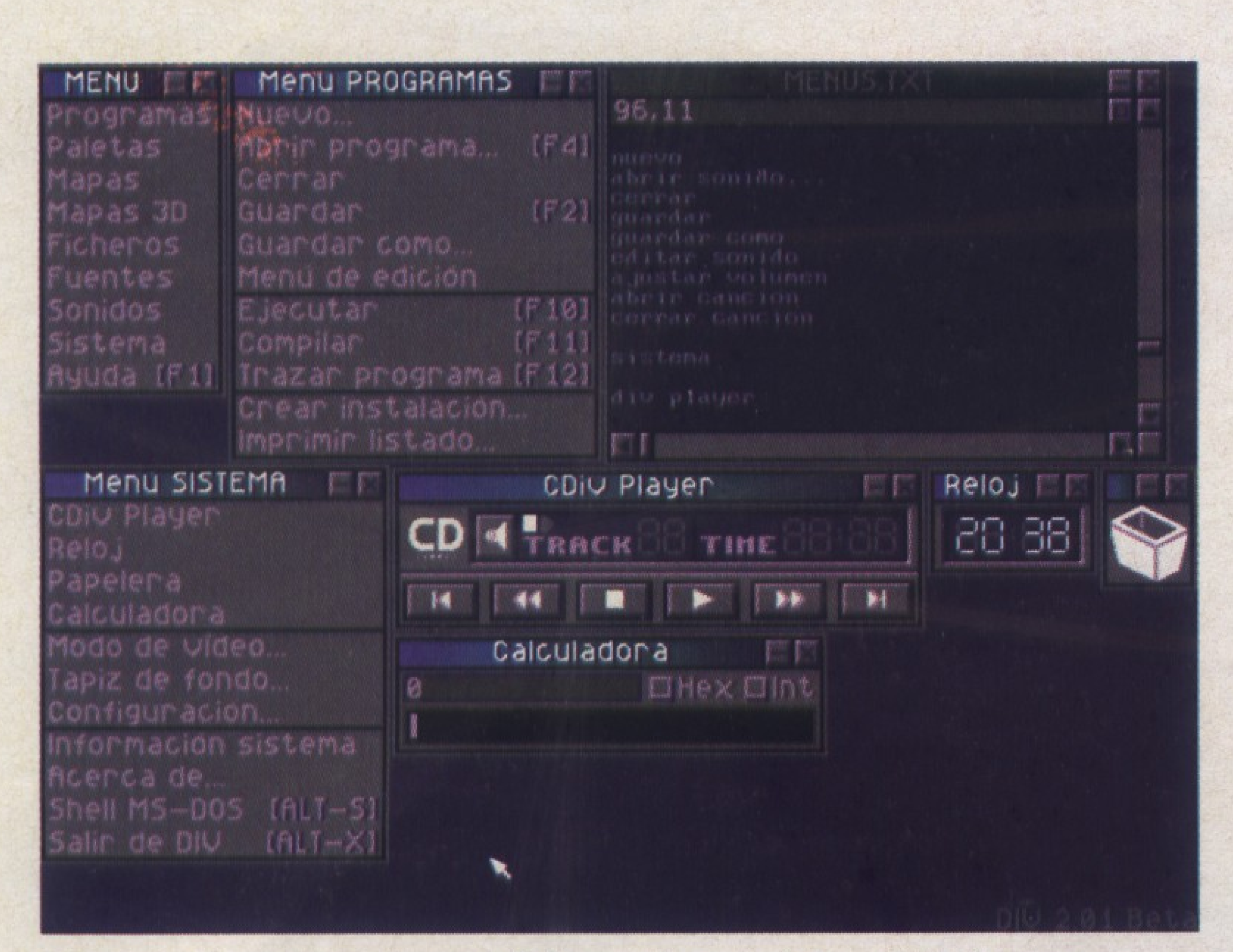
Nombre: ....

Dirección: .... **C.P.:** ....

Ciudad: .... **Edad:** .... **Teléfono:** ....

PERFIL DE TU EQUIPO: **Procesador:** ❑ 486 ❑ Pentium ❑ Pentium II ❑ Otro .... **Sistema Operativo:** ❑ MS Dos ❑ Windows 3.1 ❑ Windows 95 Windows NT ❑ Otro .... **Memoria RAM:** ❑ 8 ❑ 16 ❑ 32 ❑ Otro .... **Tarjeta Gráfica:** ❑ 1Mb ❑ 2 Mb ❑ 4 Mb ❑ Otra .... **Tarjeta Aceleradora:** ❑ 3Dfx ❑ Power VR ❑ Otra .... **Tarjeta de Sonido:** ❑ Sound Blaster compatible ❑ Gravis ❑ Otra .... **Lector de CDROM:** ❑ 2X ❑ 4X ❑ 8X ❑ Otro .... **Modem:** ❑ 14400 ❑ 33600 ❑ 55400 ❑ Otro ....

PERFIL DE USUARIO: **¿Para qué usas tu PC?** ❑ Juegos ❑ Programación ❑ Diseño ❑ Ofimática ❑ Otros
¿Dónde lo usas? ❑ Casa ❑ Trabajo **¿Cuántas horas al día de media lo usas?** ❑ 2 ❑ 4 ❑ 6 ❑ Otros
¿Haces uso de Internet frecuentemente? ❑ Si ❑ No **¿Tienes acceso a una red local?** ❑ Si ❑ No
¿Posees una consola de videojuegos? ❑ Si ❑ No **¿Cual?** ....
¿Donde has comprado este producto? ❑ Quiosco ❑ Tienda ❑ Centro Comercial ❑ Libreria ❑ Por Correo ❑ Otros
¿Por qué decidiste comprar el producto? ❑ Portada ❑ Precio ❑ Amigos ❑ Revistas ❑ Publicidad ❑ Tematica ❑ Otros
¿Te consideras buen...? ❑ Grafista 2D ❑ Grafista 3D ❑ Progr. Principal ❑ Progr. de Apoyo ❑ Músico ❑ Aprendiz
¿Qué revistas especializadas lees habitualmente? ....



No podemos descuidar el sonido del juego.

Sólo queda la opción de *ayuda*, que aunque no tiene otro menú, ha sido cambiada. El *look* general ha sido mejorado y se han incorporado botones que nos permiten *ir a la página anterior, ir a la página siguiente, ir al índice, al glosario, a la lista de funciones, imprimir la ayuda o ayuda sobre la misma ayuda.*

Con esto quedaría visto por encima todo el entorno. Ahora, pasemos al lenguaje viendo las nuevas palabras incluidas, agrupadas por la función que realizan.

El lenguaje

Se han hecho bastantes mejoras en el lenguaje, incluyendo nuevas funciones y constantes. Veamos primero las constantes, aunque algunas de ellas estén solas en su cometido. La primera de ellas es *all_drawing* que sirve como indicador para las nuevas funciones de primitivas. Otra constante relacionada con funciones es *c_m8* que indica el tipo de coordenada de *m8*. El siguiente grupo de constantes tiene una relación muy estrecha. Se han incorporado opciones de compilación dentro del lenguaje, que son procesadas antes de la ejecución del programa. Estas constantes, que definen los distintos tipos, son: *_max_process, _extended_conditions, _simple_conditions, _case_sensitive, _ignore_errors, _free_sintax, _no_check, _no_strfix, _no_optimization, _no_range_check, _no_id_check y _no_null_check.* Manejan aspectos como la sintaxis del lenguaje, la detección de errores, o el número de procesos.

> Para el manejo de búsquedas de caminos disponemos de tres funciones, llamadas path_find(), path_line() y path free()

Las siguientes constantes tienen que ver con el manejo de ficheros, y son: *seek_set, seek_cur y seek_end.* Sirven para situarse dentro del fichero usando las funciones que más adelante veremos. También están relacionadas con ficheros, las funciones *_hidden, _system, _volid y _subdir.* Aunque también tienen que ver con el sistema.

Y ya, dentro del aspecto de sonido, nos encontramos con las siguientes constantes: *fast_mixer, quality_mixer, sound_bits_8 y sound_bits_16*, que permiten seleccionar el tipo de calidad deseada. Con esto quedarían vistas las constantes y pasaríamos a las variables globales.

Algunas de las estructuras globales han sufrido cambios y se incluyen además estructuras nuevas. Pero vayamos poco a poco, empezando con una vieja conocida, la estructura *setup*, que tiene algún campo nuevo. La lista completa es: *card, port, irq, dma, dma2, master, sound_fx, cd_audio, mixer, rate y bits.*

Una estructura nueva es la estructura *net*, que tiene los siguiente campos: *dispositivo, com, velocidad, teléfono, cadena_inicio, tonos, servidor, max_players y num_players.* Trabaja aspectos de conexión de ordenadores, siendo posible la conexión por cable, modem o red local.

Otra estructura novedosa es la que trabaja con los nuevos mundos 3D, cuyo nombre es *m8*. Dispone de los siguientes campos: *z, camera, cámara, height y angle.* Estos campos son muy parecidos a los usados en los *modos 7*, pero con algunas mejoras.

Y hablando de mejoras, una muy notable es la posibilidad de manejar los ficheros desde disco. Existen una serie de estructuras que, junto con algunas funciones, nos permiten cargar, grabar o crear archivos en el disco. La primera de estas estructuras es *dirinfo*, que dispone de dos campos llamados: *files y name.* La otra estructura tiene más campos, su nombre es *fileinfo*, y estos campos son: *fullpath, drive, dir, name, ext, size, day, month, year, hour, min, sec y attrib.* Con estas dos estructuras se podrá obtener toda la información de cualquier archivo.

Algo notable que también se ha hecho dentro de la herramienta es el manejo de los modos *VESA*. Ahora se dispone de una estructura llamada *video_modes*, que nos permite detectar todos los modos *VESA* incluidos en cada tarjeta. Esta estructura dispone de tres campos llamados: *ancho, alto y modo.* Con esta opción se puede configurar una estructura distinta para cada tipo de tarjeta, y ordenador distinto.

Y con esto, terminamos con las nuevas estructuras, pasando a las variables, tanto globales como locales. Dentro de las globales existe una mezcolanza que no permite agruparlas de ninguna forma, por eso daremos la lista completa: *fps, argc, argv, channel, vsync, draw_z, y num_video_modes.* Dentro de las locales, todas ellas tienen que ver con los *modos 8*, la lista es la siguiente: *radius, m8_wall, m8_sector, m8_nextsector y m8_step.*

Pasamos a las funciones que, debido a su gran número y a la falta de espacio, hemos agrupado por finalidad. Así, únicamente enumeraremos el grupo, y explicaremos qué función desempeña.

El primer grupo es el que se ocupa del aspecto sonoro. Permite ejecutar tanto sonidos como canciones. La lista completa de nuevas funciones es la siguiente: *unload_wav(), load_wav(),load_song(), unload_song(), song() , stop_song(), set_song_pos(), get_song_pos(), get_song_line(), is_playing_sound(), is_playing_song() y change_channel().*

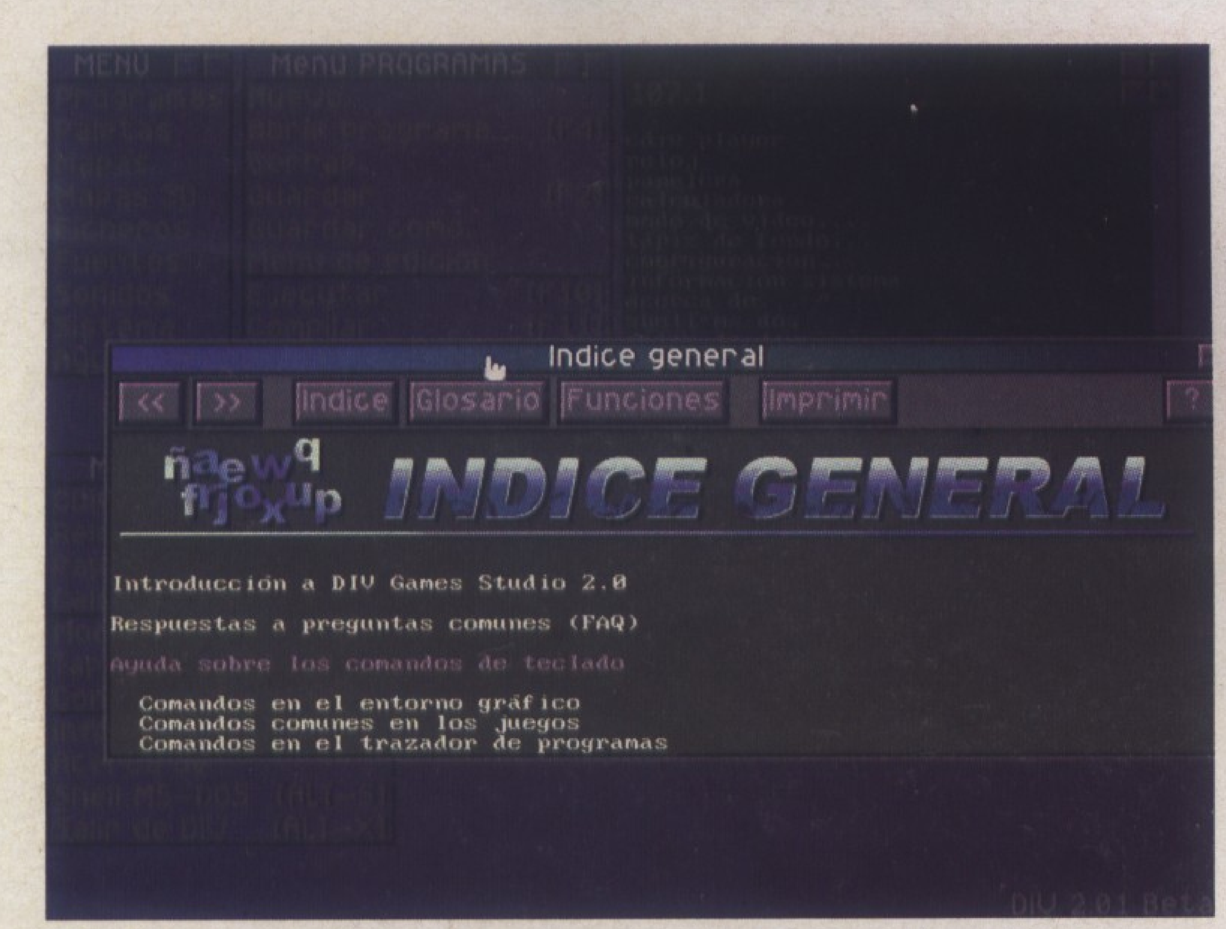
Para el manejo de red sólo se usan dos funciones, llamadas *net_join_game() y net_get_games().* Decir que también se dispone de una estructura que ya ha sido comentada.

Una función que no se ha podido meter en ningún grupo es *xadvance().* Ésta es una ampliación de la función *advance*, permite avanzar en otro ángulo distinto al asignado a la variable local *angle.*

Y hablando de moverse, para el manejo de búsqueda de caminos, disponemos de tres funciones llamadas *path_find(), path_line() y path_free().* Con ellas y los mapas de durezas creados, podremos programar de manera tan inteligente como para rebasar obstáculos.

Dentro del aspecto gráfico, también ha habido muchas mejoras. Hemos agrupado en esta sección todas aquellas que tienen algo que ver con los mapas gráficos o con las paletas de colores. La lista completa es la siguiente: *load_pcx(), unload_pcx(), new_map(), screen_copy(), save_pcx(), set_color(), save_map(), write_map(), find_color(), load_screen() y force_pal().* Muchas de ellas evitarán dolores de cabeza al programador.

Una de las mayores novedades en DIV 2 es la posibilidad de los *modos 8*. Por eso, no es raro que se



El entorno usa el mismo sistema de ventanas.

disponga de bastantes funciones que manejen este nuevo aspecto. La lista es la siguiente: *load_wld(), start_mode8(), stop_mode8(), go_to_flag(), set_sector_height(), get_sector_height(), set_point_m8(), get_point_m8(), set_fog(), set_sector_texture(), get_sector_texture(), set_wall_texture(), get_wall_texture() y set_env_color()*. Con todas ellas podremos crear mundos 3D además de modificarlos en tiempo de ejecución.

Un aspecto que fue dejado bastante de lado en la antigua versión fue el manejo de *cadenas de texto*. Esto no ha pasado en la nueva, y se disponen de numerosas funciones que nos ayudan al tratamiento de textos. Se han incluido nuevos tipos de datos en el lenguaje para un mejor manejo de todo tipo de variables. Las funciones relacionadas con los textos son: *char(), strcpy(), strcat(), strlen(), strcmp(), strchr(), strstr(), strset(), upper(), lower(), strdel(), calculate() y itoa()*.

Otra función que se ha quedado sin agrupar debido a su finalidad es *qsort()*. Con ella podremos ordenar todo tipo de datos, y será muy útil para la tabla de *records* o programas que tiren hacia la gestión.

El siguiente grupo trata sobre la gestión de ficheros. Este aspecto también era un poco parco en la antigua versión. En ésta, ha sido mejorado con la inclusión de todas estas nuevas funciones: *fopen(), fclose(), fcloseall(), fread(), fwrite(), fseek(), ftell(), filelength(), flush(), get_dirinfo(), get_fileinfo(), getdrive(), setdrive(), chdir(), mkdir(), remove() y disk_free()*. La gente que haya programado en C estará familiarizado con muchas de ellas, teniendo en este caso la misma finalidad.

Existen algunas funciones que nos permiten trabajar con el sistema de manera más óptima. Este grupo de cuatro elementos tiene que ver con el manejo de memoria, aunque también se le ha añadido una función de sistema. La lista es: *memory_free(), ignore_error()malloc() y free()*

Para todos los matemáticos se ha incluido un grupo de funciones relacionadas con la trigonometría: *sin(), cos(), tan(), asin(), acos(), atan() y atan2()*. Como se puede comprobar, cada función describe su cometido.

Ahora también es posible dibujar círculos y cajas en pantalla. Para ello se disponen de tres funciones llamadas *draw(), delete_draw() y move_draw()*. A este tipo de dibujos se les ha dado el nombre de *primitivas*.

Por último, DIV 2 ha querido rizar el rizo y ha añadido una serie de funciones totalmente novedosas: *encode(), encode_file(), decode_file(), compress_file() y uncompress_file()*. Lo que permite encriptar y comprimir ficheros de cualquier tipo.

Dando un repaso general a todas las funciones, lo primero que salta a la vista es que tocan numerosos campos. Es decir, que con el uso de dichas funciones, lo mismo se puede hacer una aplicación multimedia, que un programa de gestión o un juego. Y ya que para esto último estaba realizado DIV, también se ha optimizado este campo, con la inclusión de funciones más afines.

Conclusión

Una vez dado un paseo por toda la nueva herramienta, daremos nuestra opinión al respecto. En general, se han hecho bastantes mejoras. Para comprobarlo, solamente tendríamos que contar el número de ellas llegando a las tres cifras. Algunos de los cambios realizados, son una mejora notable en la creación de videojuegos.

Está claro que le faltan algunos elementos que ha sido imposible incluir, como el aspecto del color; eso sin contar que es MS-DOS. También sabemos que se ha quedado alguna idea en el tintero por falta de tiempo.

Resumiendo, el veredicto ante la pregunta de si es rentable comprarse DIV 2 por las nuevas mejoras es a favor. Es evidente que cada cual podrá tener su propia visión de esta herramienta; lo que realmente se ha pretendido en el artículo es enumerar las nuevas mejoras. Para los que no lo tengan muy claro, queda esta opinión, DIV 2 ha cambiado tanto respecto a DIV 1 que merece la pena rascarse el bolsillo.

Nada más, espero veros con la nueva herramienta ya en marcha en el próximo número. Así, sí que tendréis una visión real de DIV 2, y hasta incluso ya habréis creado vuestros propios videojuegos usando las nuevas opciones.

Antonio Marchal

DIV 2: Fecha de salida

Ha habido cierto revuelo con la fecha de salida de la nueva herramienta. Actualmente está en fase de *Beta-testing* y no ha cumplido con las primeras fechas de salida por problemas de ultima hora.

Todos estos problemas han sido resueltos, estando la herramienta totalmente acabada. Pedir a todos los anhelantes usuarios un poco de paciencia. Pensar que esta espera ha sido necesaria para arreglar algunos aspectos fundamentales de la herramienta y que estará muy pronto a la venta.

El que escribe no quiere dar una fecha exacta de salida del producto, aun así, sí se podría decir que, estando el producto en fase de últimas pruebas, la salida a la calle es inminente. Además, creo que la espera merece la pena.



Dinamic

Una compañía con solera

Una de las más veteranas compañías del sector se mantiene al pie del cañón tras muchos años sorprendiendo a los usuarios. La compañía se encuentra ahora en una fase de madurez en la que están surgiendo algunos de los más famosos productos del software de entretenimiento salido de nuestras fronteras.

Muchos años lleva Dinamic dedicada al mundo de los videojuegos. Era ya uno de los baluartes del software patrio allá en los lejanos ochenta, cuando la industria apenas había empezado a surgir. En el campo de los PCs también se incorporó desde el principio, y fue una de las compañías que inundó nuestros kioscos con productos que no tenían una calidad demasiado elevada, pero poseían un precio muy bajo. Muchos de nosotros hemos jugado con sus productos a lo largo de años.

Entonces no tenían una calidad excesiva, y la única ventaja sobre los productos venidos de fuera era su bajo precio. De un tiempo a esta parte, sin embargo, la situación del software lúdico español ha mejorado considerablemente y a este carro también se ha sumado Dinamic. Perfectamente reconocible por uno de sus productos estrella, PC Fútbol, una auténtica bandera de la compañía, no sólo se ha decidido a mejorar considerablemente la calidad de los productos realizados en casa, sino que nos ofrece una serie de productos traídos del extranjero, y de compañías de la calidad de Empire o Microïds.

Así pues, muy variada es la oferta que pone a disposición de los usuarios en estos momentos. La producción propia está aumentando con un buen número de nuevos proyectos que, en un plazo no mucho mayor de un año, empezarán a llegar a los usuarios. Y, por el lado de los productos traídos de fuera, tienen una enorme ventaja sobre los de otras distribuidoras, y es que mantienen el bajo precio que caracteriza los lanzamientos de la compañía. Echemos un vistazo ahora a algunos de los productos que en estos momentos se encuentran en el mercado.

PC Fútbol 7

Lleva ya varios meses a disposición de los lectores y en realidad poco se puede decir de él, pues es uno de los programas realizados en España más conocidos entre los aficionados a los deportes. En esta ocasión el juego gira en torno a la temporada 98-99. Como sabéis, este título ofrece tanto opciones de mánager como de simulación, además de un amplio abanico de posibilidades que nos limitaremos a enumerar: Anuario, Historia, Base de datos, Infofútbol y ProQuinielas. Otras opciones menores complementan uno de los títulos deportivos más completos que hay.

Además, la presentación del producto es muy buena. Viene acompañado por un extenso manual en el que es posible encontrar fichas de todos los jugadores de primera división encuadrados en sus respectivos equipos. Además de tener la licencia oficial de la Liga de Fútbol Profesional, cuentan con la ya clásica participación del comentarista de fútbol de Canal + Michael Robinson, que pone su especial forma de contar las cosas.



Apache Havoc

Una de las aportaciones de Empire es este simulador de helicópteros en el que puedes tomar los mandos de uno de los dos que aparecen en el título. La compañía es experta en este tipo de simulaciones y, de hecho, ofrecen otro título similar que comentamos más abajo. El programa posee las virtudes típicas de los programas del género, pero también sus defectos. La calidad gráfica durante el desarrollo del juego es considerable, pero los menús aparecen un poco parcos, algo habitual en este tipo de juegos.

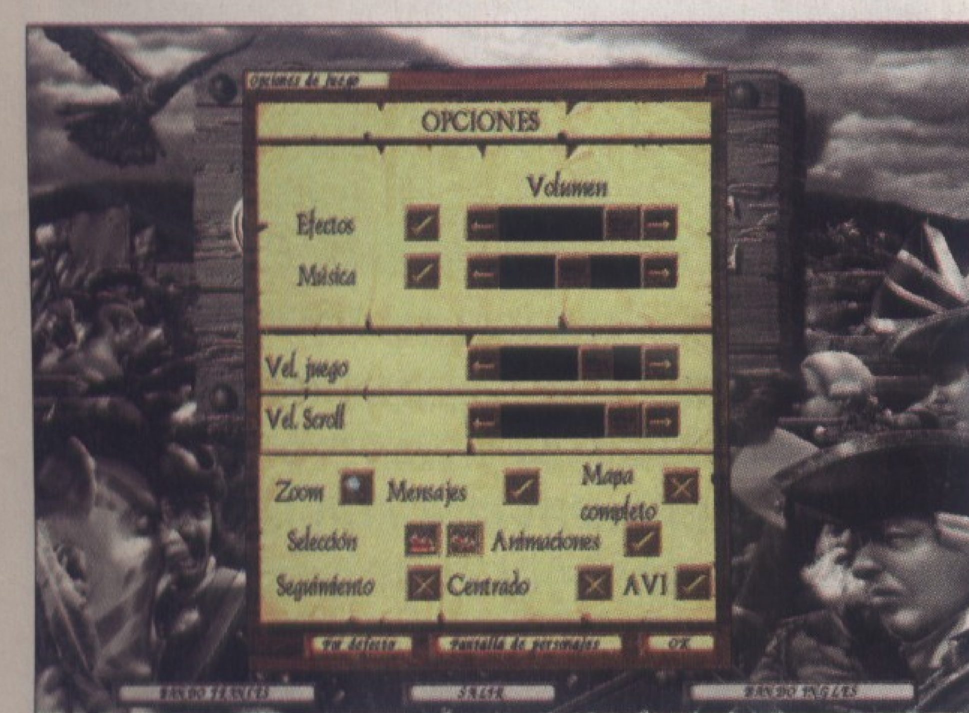
La dificultad es elevada para un usuario no avisado. Por ello, es muy importante leer a fondo el extenso manual que acompaña al producto. Sin él, tienes la sensación de estar un poco perdido en un entorno desconocido. El realismo de los helicópteros y de sus respectivas cabinas es considerable, y en igual medida es complejo el manejo del teclado. Sin embargo, en el manual han introducido una plantilla del teclado en la que se incluyen todas las funciones de las teclas.



Shogo: un excepcional arcade capaz de hacer sombra a grandes del género.

Fields of Fire

Se trata de otro de los productos de Empire que nos ofrecen, esta vez con el sabor de la más clásica estrategia. Después de haber demostrado que dominaban el género con su anterior *Tribal Rage*, los chicos de Empire se han lanzado con un programa que, por su temática, tiene quizá más cabida en Estados Unidos que en España. Sin embargo, se trata de un juego que ofrece una calidad considerable de manos de un aparato gráfico excelente. Con el sobrenombre de *War along the Mohawk*, el programa se sitúa en el año 1757, en la época de las guerras franco inglesas por el control de las tierras de la frontera entre Canadá y el Este de Norteamérica.

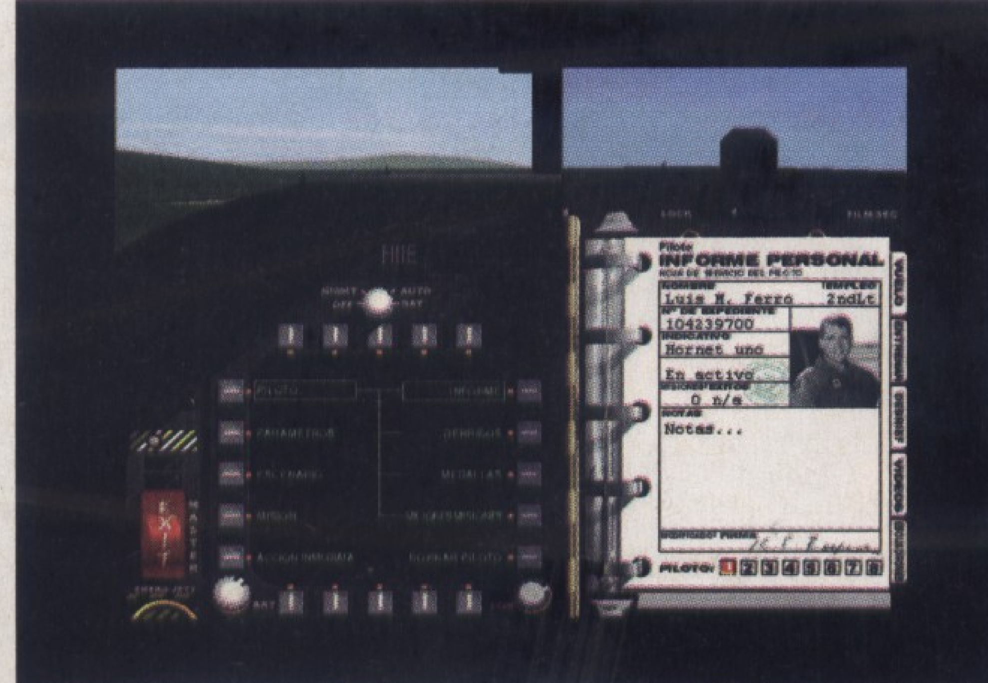


Fields of Fire: excelente estrategia en la indómita Norteamérica.

El programa cuenta con algunos elementos de rol, como la posibilidad de ir aprendiendo nuevas habilidades; además, ofrece elementos novedosos, tales como la captura de oficiales enemigos, el rescate de prisioneros, etc. Teniendo en cuenta su bajo precio, es un producto muy interesante dentro de su género.

F/A 18 Korea

Otro simulador de Empire, en esta ocasión de caza de combate. Se trata, por si fuera poco, de una edición especial centrada en las fuerzas aéreas españolas. El juego, que requiere aceleración 3D como sucedía con el simulador de helicóptero comentado más arriba, nos permite tomar los mandos de un F/A 18 en unas complicadas misiones en la guerra entre Corea del Norte y Corea del Sur. Detrás del programa se esconde un reconocido veterano de los altos vuelos que todavía realiza vuelos comerciales.



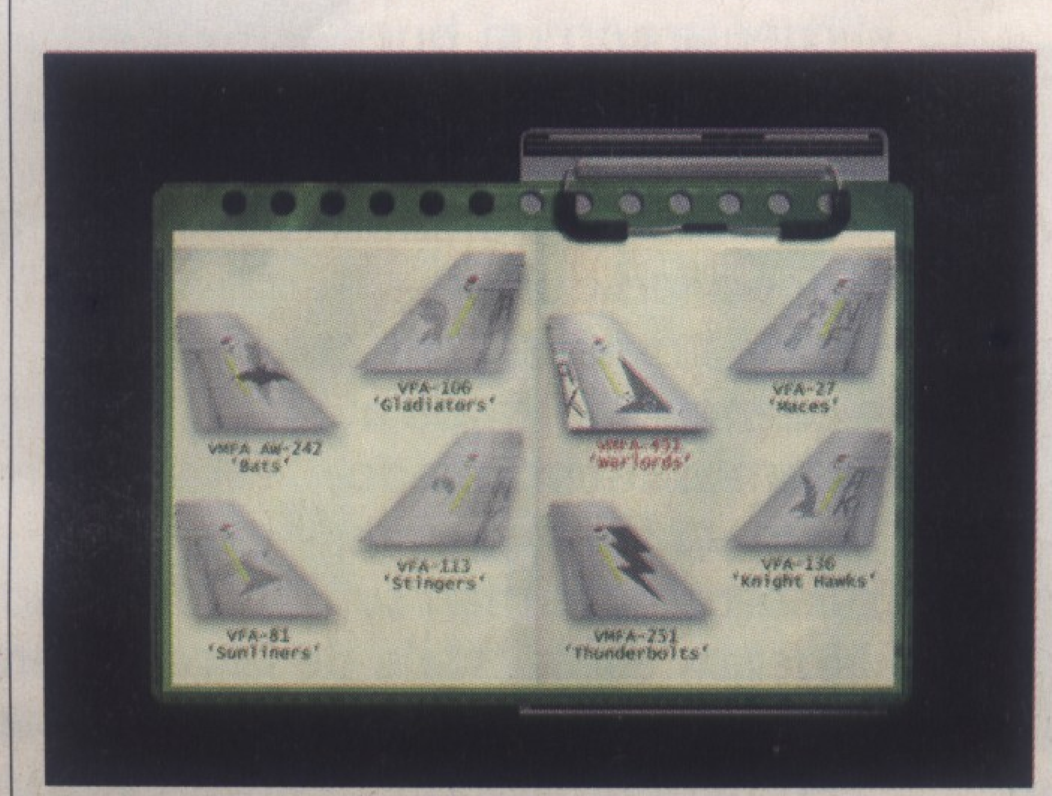
F/A-18: un simulador muy atractivo y con muy buena presentación.

El programa, nuevamente, posee gran calidad gráfica, así como una considerable complejidad. Sólo disfrutarán de él los expertos jugadores de estos juegos, pues los demás se verán un poco perdidos ante el elevado número de controles, a pesar de la plantilla con las funciones del teclado que acompaña al producto. Y a pesar del manual, muy extenso y completo, de acuerdo con las necesidades del programa.

Shogo

Llegamos, por último, a uno de los más excepcionales lanzamientos de la compañía. Arcade 3D, al estilo de *Quake II* pero con alguna que otra novedad muy interesante. Es otro producto importado pero esta vez, nos llega por insospechados caminos, a través de Microïds y antes de Monolith. Y, curiosamente, es un programa con un *look manga* maravilloso. Tienes que asumir el papel de un soldado que ha perdido a su familia a manos de unos grupos terroristas, y debe cumplir una serie de misiones destinadas a acabar con el enemigo.

El programa, en principio, requiere aceleración 3D, aunque se puede ejecutar en ordenadores P233 o superiores sin necesidad de la mencionada tarjeta. En cualquier caso, la calidad es excelente, la mejor sin duda de todos los programas que comentamos en este reportaje. Además, en el programa abundan pequeños detalles de calidad: objetos y armas escondidos en camiones y cajas, vídeos que irrumpen con la llegada de algunos enemigos, etc. Y todo por menos de 3.000 pesetas. Una maravilla de programa, lo mires por donde lo mires.



Los menús de F/A 18 responden a lo habitual dentro de este género.

Cómo hacerse millonario programando

Nuevas oportunidades, grandes promesas

La industria del ocio está en alza, al igual que todo lo relacionado con la informática, así que, en este doble contexto de auge, no es de extrañar que los videojuegos se empiecen a tomar muy en serio, tanto por parte de los inversores, como por el público en general. Si estabas pensando vivir de píxeles y marcianos, aquí está la oportunidad que siempre has soñado para iniciar tu carrera.

Cuando se pregunta por la clave del éxito, siempre se suele salir del paso con la habitual, pero no menos cierta, "es cuestión de estar en el lugar y en el momento adecuado". Ya hemos hecho referencia en la entradilla a que el momento sí, es el adecuado. Nos encontramos ante un mercado en el que miles de personas están ya agazapadas en la parrilla de salida y, aunque la competición vaya a ser realmente dura, todo indica que aún hay sitio para muchos participantes nuevos. Pero ¿Y el lugar? ¿Tendremos en este país las mismas oportunidades que en otros? La respuesta es "no", aunque con reservas, más bien un "no todavía". Aunque haberlas, como las meigas, haylas. De hecho, nuestra generación goza de muchísimas más posibilidades que la anterior para que a un programador de videojuegos se le tome en serio. De todas maneras, tengamos presente que esto no ocurre porque se tenga en cuenta su capacidad creativa, al menos por parte del público en general, sino porque da dinero. Lo que significa que un aficionado al software lúdico, que cuente con un buen equipo de programación y una gran idea, puede tomar esto como profesión... una profesión muy rentable, por añadidura.

Un aficionado al software lúdico que cuente con un buen equipo y una gran idea puede dedicarse a esta rentable profesión, que además posee unos alicientes personales que producen satisfacción

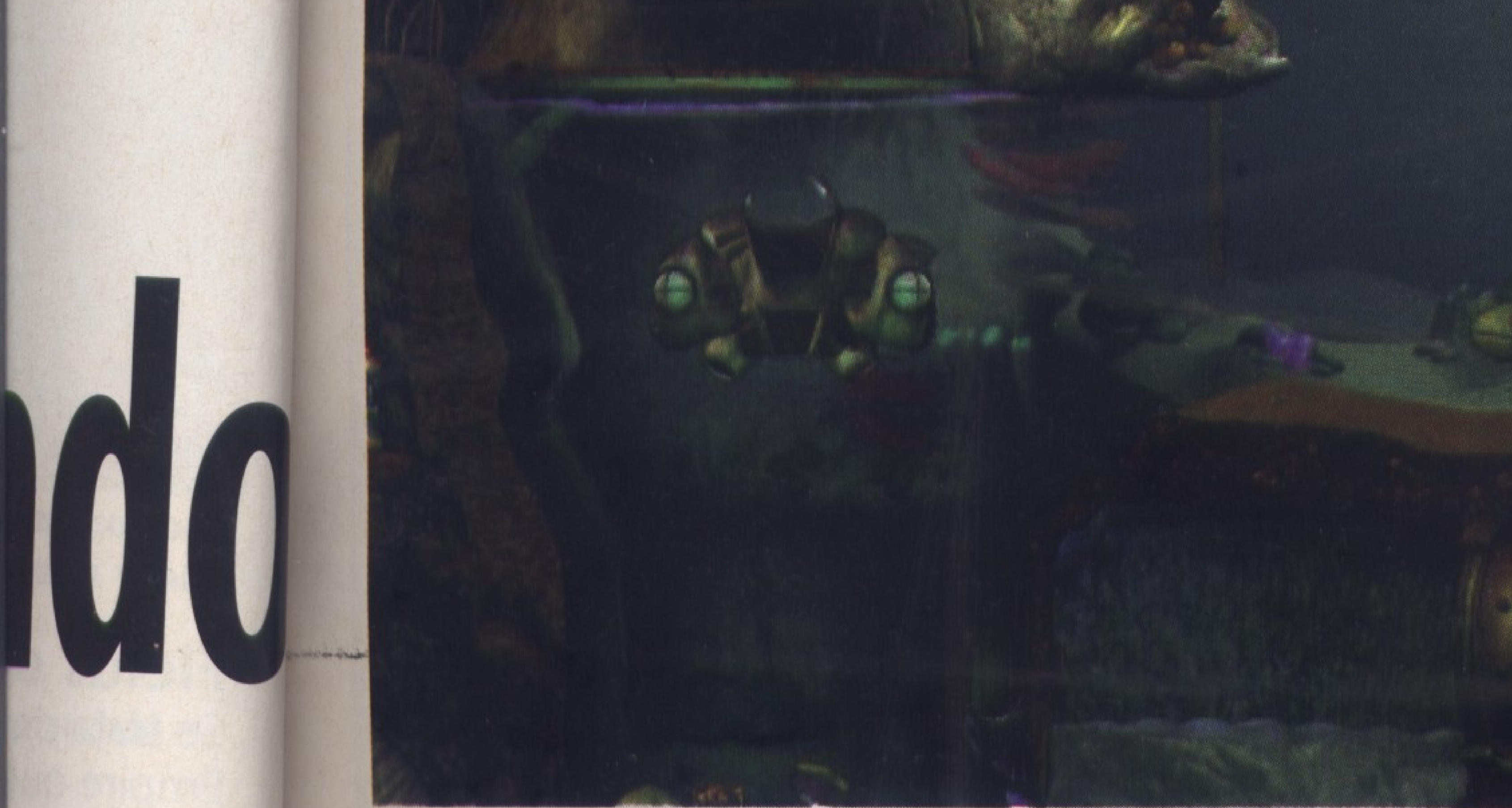


El mundo de los videojuegos es un filón que todavía no ha terminado de desarrollarse.

Esto no es más que la consecuencia lógica de algo que ya hemos mencionado. Tengamos en cuenta los siguientes factores: el ascenso que han sufrido los videojuegos en la escala de la consideración que tiene de ellos la sociedad en general, los dividendos cada vez mayores que producen, el "empujón" que son capaces de dar a industrias más serias (como es el caso del hardware con las tarjetas aceleradoras) y el interés que tienen éstas en que aquellos sigan creciendo; finalmente, hay que destacar el desarrollo de los mismos en cuanto a tecnología y concepto, razón por la que cada año atraen a un público más variado.

¿Genios o equipos?

El desarrollo de los videojuegos está entre dos frentes aparentemente opuestos. Si los miramos desde el punto de vista del mercado, es fácil encontrarnos detrás equipos muy competentes que trabajan para cumplir los requisitos de calidad que se les haya impuesto en un plazo determinado. Cada cual suele ocuparse de un apartado particular del juego que no guarda parecidos con otros del mismo. La especialización es en este caso una ventaja, puesto que saber un poco de todo sólo tiene valor a la hora de coordinar el equipo y tener una visión global y precisa de la obra. Pero si miramos este mundo desde una perspectiva artística (aunque sólo sea por la capacidad creativa



Saber dar forma a una buena idea es un negocio muy rentable.



Sid Meier, Peter Molyneux y John Romero son ya programadores de culto.

que pide un buen juego) solemos encontrarnos con nombres propios, con programadores "de culto" que a lo largo de esta corta historia han destacado, principalmente, por saber realizar una idea original. Entre ellos podemos citar al midas de la estrategia, Sid Meier; al creador de los juegos "tipo dios", Peter Molyneux o al que ha llevado la perspectiva subjetiva a lo que es ahora, John Romero.

Antes era más habitual encontrarnos con un solo desarrollador, o un equipo muy pequeño detrás de cada programa, puesto que la tecnología de los ordenadores de 8 bits no necesitaba tantos recursos. Era como hacer una pequeña obra de arte: horas detrás de un ordenador y una buena idea que

Entrevista con Fernando Pérez, programador principal de Dinamic Multimedia

Game Over: Supongamos que alguien tiene una buena idea para hacer un juego, ¿qué habría de hacer para que ese proyecto se convirtiese en un producto comercial?

Fernando Pérez: Tendría que ponerse en contacto con un grupo ya formado y exponerles la idea. Pero este camino es muy difícil en la actualidad: hay pocas empresas y no suelen apostar por ideas externas.

Game Over: ¿Cómo ves el software español de cara a dar oportunidades a futuros programadores?

Fernando Pérez: Ahora muy bien. Hay mucha demanda de grafistas, programadores, etcétera. Sólo hay que echar una ojeada en una revista para ver la cantidad de anuncios que hay al respecto.

Game Over: ¿Crees que la capacidad creativa del programador está muy limitada por las leyes del mercado?

Fernando Pérez: Veo al programador, más que como un artista, como un trabajador de una construcción que ha de adaptarse al edificio que se está levantando. Forma parte de un equipo que desarrolla un proyecto que ha de tener muy en cuenta estas leyes del mercado.

Game Over: ¿Qué cambio estás esperando, o crees que va a producirse, dentro del mundo de los videojuegos?



Fernando Pérez: No creo que vaya a haber ningún cambio significativo. En todo caso, éste se limitará a una mejora tecnológica en función de los avances en hardware, por ejemplo con la aparición de tarjetas que pinten superficies curvas. Pero, en el fondo, será como seguir agregando más efectos, detalles y tecnología en general a los antiguos juegos de Spectrum. Sencillamente como añadir más huevos a la misma tortilla.

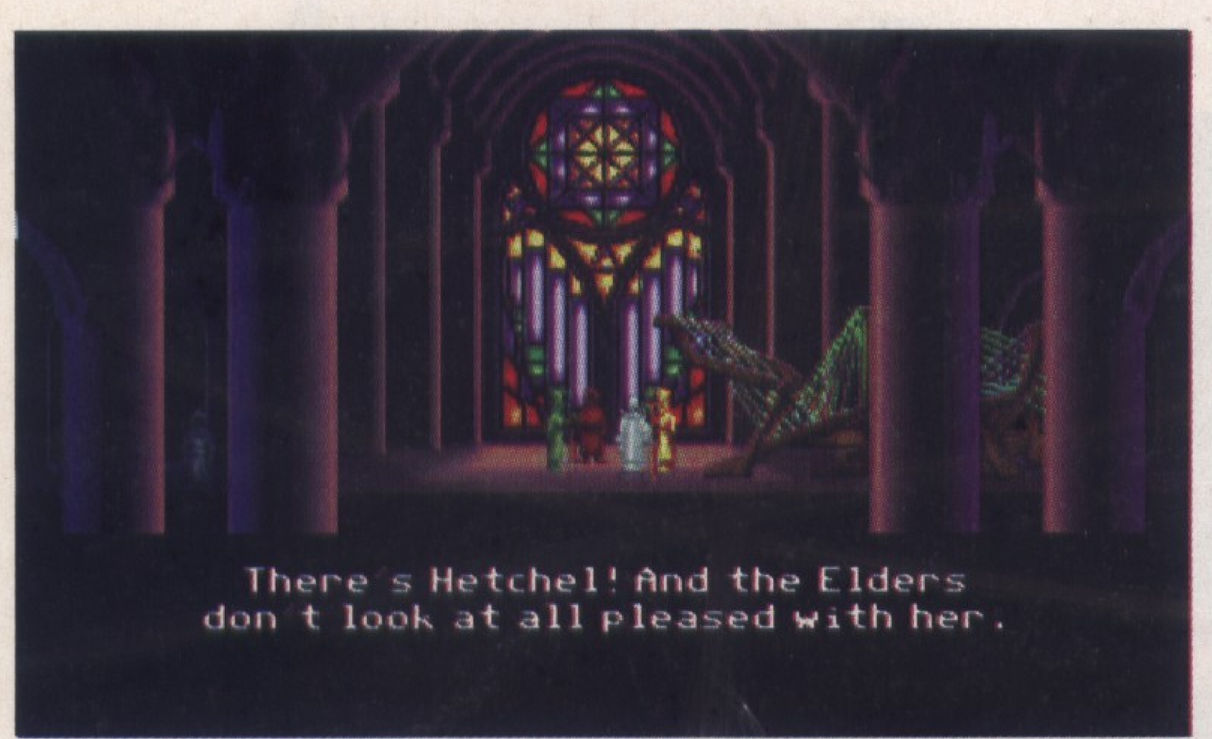
Game Over: ¿Qué es preferible, un programador que sepa un poco de todo o la especialización?

Fernando Pérez: yo prefiero un grupo de muchas personas especializadas.

Game Over: ¿Has querido ser lo que eres ahora? Es decir, cuando eras un niño ¿convertirte en un programador de videojuegos es lo que querías ser de mayor?

Fernando Pérez: No sé si desde que era niño, pero desde los doce años, que es cuando empecé a interesarme por este mundo, sí quería ser lo que soy ahora.





Internet ha abierto nuevos horizontes con los partidos multijugadores.

encajase en un mercado todavía virgen. Sin embargo, no había alcanzado esta industria la suficiente popularidad para que esos nombres fueran más allá de los títulos de créditos del propio juego. Es difícil determinar el momento en el que se produjo el cambio que sacaría del anonimato a estos creadores. Una hipótesis sería que fue en el tiempo en que convivieron los primeros compatibles con máquinas como los Spectrum. La nueva tecnología trajo consigo un género desconocido hasta entones: las videoaventuras, programas que servían de medio para contar una historia. Más creatividad, tanto a la hora de adaptar el concepto de los videojuegos en sí a las nuevas plataformas, como para reinventar los géneros, combinarlos o modificarlos de tal manera que pudieran servir de base a otros completamente nuevos. En este contexto, saber dar cuerpo a una buena idea era un negocio muy rentable, máxime cuando las distribuidoras buscaban ese toque de seriedad que podía otorgar el nombre del programador en la portada de un juego, como si de un libro se tratase. Un ejemplo que ilustre estas líneas es el caso de Brian Moriarty y su mítico *Loom*.

Anticipar el futuro es tan sencillo como tomar lo que en el presente tiene éxito y saber prolongarlo con cierta lógica

A la hora de dar los primeros pasos en el software de entretenimiento, un futuro programador debería pensar ya en el papel que va a escoger. Si sus conocimientos se limitan a un área de la informática o piensa que ya está todo descubierto y que sólo queda perfeccionar lo que ya existe, entonces su puesto estaría en un grupo de desarrolladores. Sin duda, es la opción más accesible, además de ser una escuela de formación ineludible. Por supuesto, los ingresos tendrán una relación directa con el éxito del grupo en su conjunto. En cambio, si su capacidad intuitiva le ha revelado un terreno aún sin explorar y tiene los suficientes conocimientos como para coordinar un grupo de programadores y transmitirles correctamente su idea, tal vez esté en la situación de poder empezar a soñar con que algún día su nombre ocupará más espacio que el título de su juego. Ahora bien, esto es casi imposible sin pasar antes por aprender junto a un grupo, entender las tendencias del mercado y saber proyectar su mente hacia el futuro (en esto de la creatividad, siempre hay que ir por delante). Opción más difícil, pero que a la larga es, hoy por hoy, la única manera de hacerse millonario en esta industria.

Pensando en el futuro

Internet nos ha acercado las partidas multijugador. Todavía quedan cosas por hacer, como perfeccionar los modos y formas de conexión (en el apartado de la tecnología) e inventar modalidades de juego que sepan distinguirse bien de los tradicionales *deathmatch* y partidas cooperativas. Además está surgiendo, cada vez con más fuerza, el concepto de "mundos permanentes". Esto es, escenarios virtuales en continua evolución, ya sea por su propia "naturaleza artificial" o por el hecho de guardar las acciones de la multitud de jugadores que los habitan. El ejemplo más característico es *Ultima Online*, aunque ya hay varios títulos que le siguen a la zaga, como *Mankind*.



Muchos apartados quedan todavía por explorar como la inteligencia artificial.

Otro apartado que queda por explotar es el de la inteligencia artificial. Dentro de poco no nos conformaremos con aniquilar a personajes que posean una inteligencia preprogramada, sino que querremos verlos aprender, evolucionar (como un tamagotchi pero a lo grande) en fin, que puedan proponernos nuevos y mejores retos en cada partida. Imaginaos las posibilidades de jugar a *Quake* con *bots* que aprendan tanto como los animalitos de *Creatures*, o que se pueda adquirir oponentes no humanos con memoria a los que podamos enseñar a jugar a nuestro juego de estrategia favorito...

Anticipar el futuro es tan sencillo como tomar lo que en el presente está teniendo éxito y saber prolongarlo con cierta lógica. Esto nos dará una buena idea que, junto a un buen grupo de programadores y los conocimientos necesarios, puede ser el inicio de hacer de tu hobby favorito la más rentable de las profesiones. Sólo un consejo más: mira la evolución del mercado, no dejes que tus propios gustos te impidan conocer qué es lo que más demanda el público, al fin y al cabo, ellos habrán de comprar el día de mañana tu producto... a no ser que consideres que eres capaz de convencer con tus ideas.

Ignacio Pulido

Más que palabras, cifras

La siguiente información, parte de la cual se ha obtenido de un informe público de PC Data, os puede dar una idea acerca del impresionante crecimiento que se está llevando a cabo en el mundo de los videojuegos. Además, tal como mencionamos en el artículo, es una orientación para saber qué tipo de juegos tendrán una mayor aceptación en el futuro y qué géneros se están consolidando.

- La venta de videojuegos ha aumentado un 13% durante 1998.
- Los precios han bajado en general un 8'9%. Si comparamos esto con el anterior dato, obtenemos que cuantos más juegos originales se venden, más bajan los precios... aunque sea una rebaja lenta todavía.
- La compañía que más ha vendido, un 19% del mercado, ha sido Havas (anterior Cendant Software y Coktel en España). Le sigue Electronic Arts con un 13% y GT Interactive con un 9%.
- La última entrega de la saga de *Zelda* ha recaudado más dinero que la película "Bichos".
- El triunfador del año 1998 ha sido StarCraft, con más de un millón de unidades vendidas.
- Al día de hoy, el juego más vendido en todo el mundo es *Baldur's Gate*.

Entrevista con Gonzalo Suárez y Javier Arévalo, de Pyro Studios

Game Over: Supongamos que un joven programador tiene, o creer tener, una muy buena idea para hacer un juego. ¿Nos podíais explicar qué pasos ha de seguir para que esa idea se convierta en un programa comercial?

Gonzalo Suárez: Es una pregunta muy general. En principio habría que ver de qué manera quiere ese presunto programador incorporarse a este mundillo, si desea adherirse a un grupo ya formado o si pretende sacar adelante un proyecto propio. Si se trata de esto último, sería conveniente que recibiese el asesoramiento de alguien con experiencia, que le pudiera dar una orientación si esa idea es o no viable. En caso de que lo fuera, habría de ponerse en contacto con gente que fuera capaz de desarrollarlo, con productores. Pero, en principio, tiene que tener claro que una idea en sí vale muy poco en este mercado. Ha de estar respaldada con un prototipo que de solidez a la misma.

Game Over: ¿Cómo veis el panorama del software de entretenimiento en España de cara a dar oportunidades a los nuevos programadores?

Gonzalo: La verdad es que muy limitado. Sobre todo porque no hay sitios donde poder formarse en esta materia.

Game Over: La capacidad creativa ¿está muy limitada por las leyes del mercado?

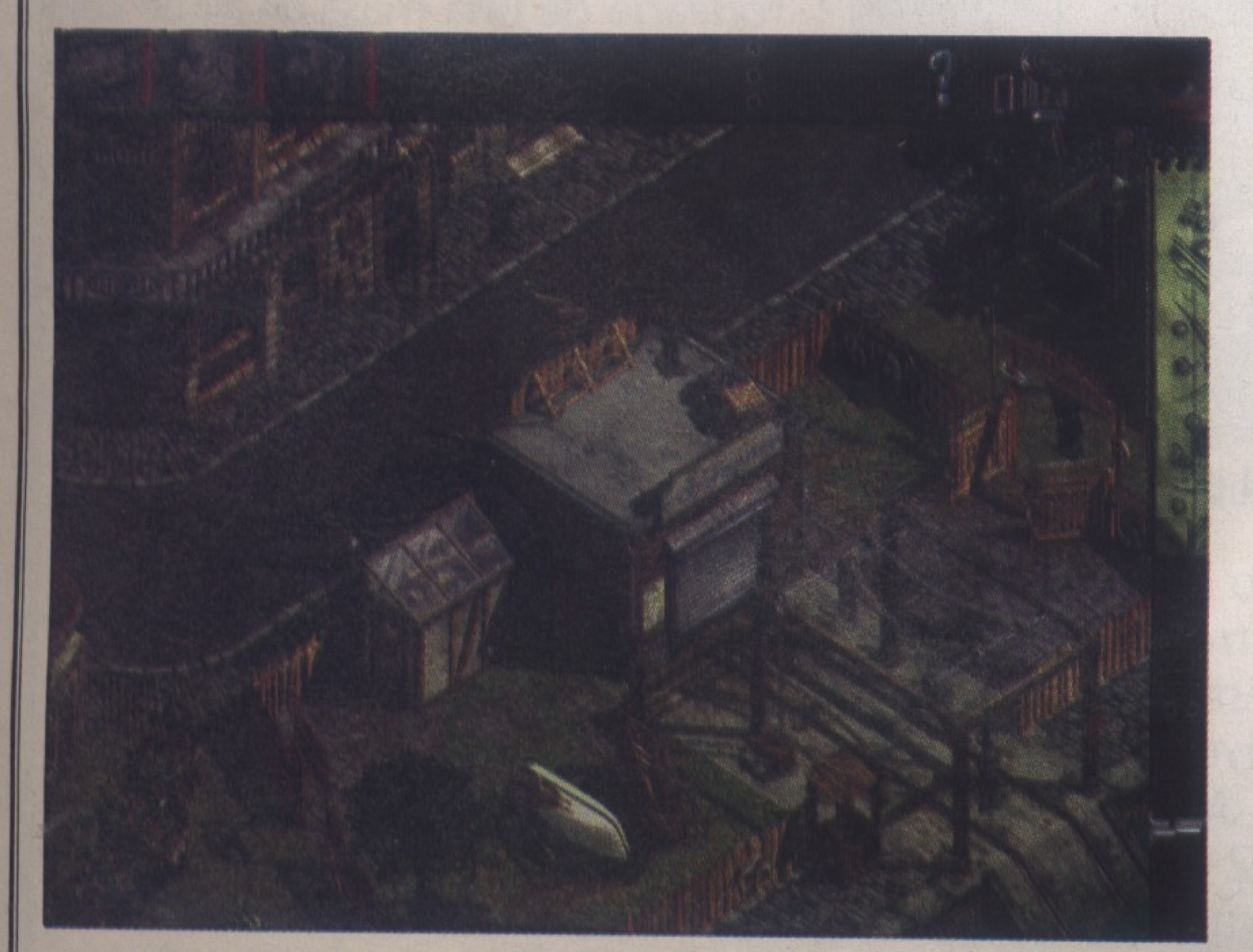
Javier Arévalo: Lo que realmente importa en este caso es la confianza que el programador tenga en su juego y si es capaz de plasmar esa idea en un programa; es decir, si puede llevarla a cabo. Precisamente es esa desconfianza la que pone límites a la capacidad creativa. A veces, uno se desvía de la idea original precisamente para adaptarla a lo que piensa que el mercado va a demandar...

Gonzalo Suárez: Lo que necesita es un producto hecho con rigor que respalde esa idea, esa capacidad creativa.

Game Over: A vuestro juicio ¿qué es lo más importante de un videojuego?

Gonzalo Suárez: Que una vez vendido, seas capaz de recomendarlo. Que satisfaga al usuario. Con respecto a éstos, hay muchas clases diferentes, así que unos se decantarán por los gráficos, otros por la originalidad, etcétera. Por mi parte, lo que más me interesa es la jugabilidad del mismo.

Javier Arévalo: Es muy importante que haya un equilibrio entre todos los elementos que lo componen, que no se potencie un aspecto en detrimento de otros. Personalmente, me parece especialmente atractivo el apartado tecnológico, pero no sacrificaría la jugabilidad por el mismo.



Game Over: ¿Qué cambio importante estáis esperando que ocurra o pensáis que ocurrirá en el mundo de los videojuegos?

Gonzalo Suárez: Espero un cambio en la industria. Que sea más madura, seria. Este cambio será lo suficientemente relevante para que los videojuegos den un gran vuelco. También se seguirán potenciando las opciones de multijugador, así como la inserción del juego en un entorno dramático, que haya detrás una historia.

Javier Arévalo: Aparte de lo que ha dicho mi compañero, quería añadir una cosa. La evolución de los videojuegos es probable que pase, o sería interesante que pasase, por recrear personajes inexistentes con cierta carga, con voluntad propia, como si tuvieran vida.

Gonzalo Suárez: Hay una forma de ilustrar este concepto. Supongamos que jugamos a *Quake*. Este es un buen ejemplo de juego sin una historia. Pongamos que utilizamos bots (muñecos programados que simulan partidas multijugador), pero que no se limiten a tener un programa determinado por el cual aprenden a moverse por un mapa o simulan tus rutas, sino que son capaces de almacenar cierta experiencia cada vez que los utilizamos. Terminarían por convertirse en algo con cierta conciencia, actuarían como si estuvieran vivos, plantearían un reto en cada partida al tiempo que se iría creando una historia, duelo a duelo. De este modo nacería una relación entre el jugador y esa inteligencia artificial... no sería extraño que acabáramos enfadándonos con ellos o incluso insultándolos, como si de un jugador humano se tratase.

Game Over: Un concepto interesante es el de los llamados "mundos permanetes". Es decir, entornos creados en la Red que seguirían desarrollándose, gracias a la inteligencia artificial o a la manipulación de otros jugadores, aunque nosotros desconectásemos el ordenador.

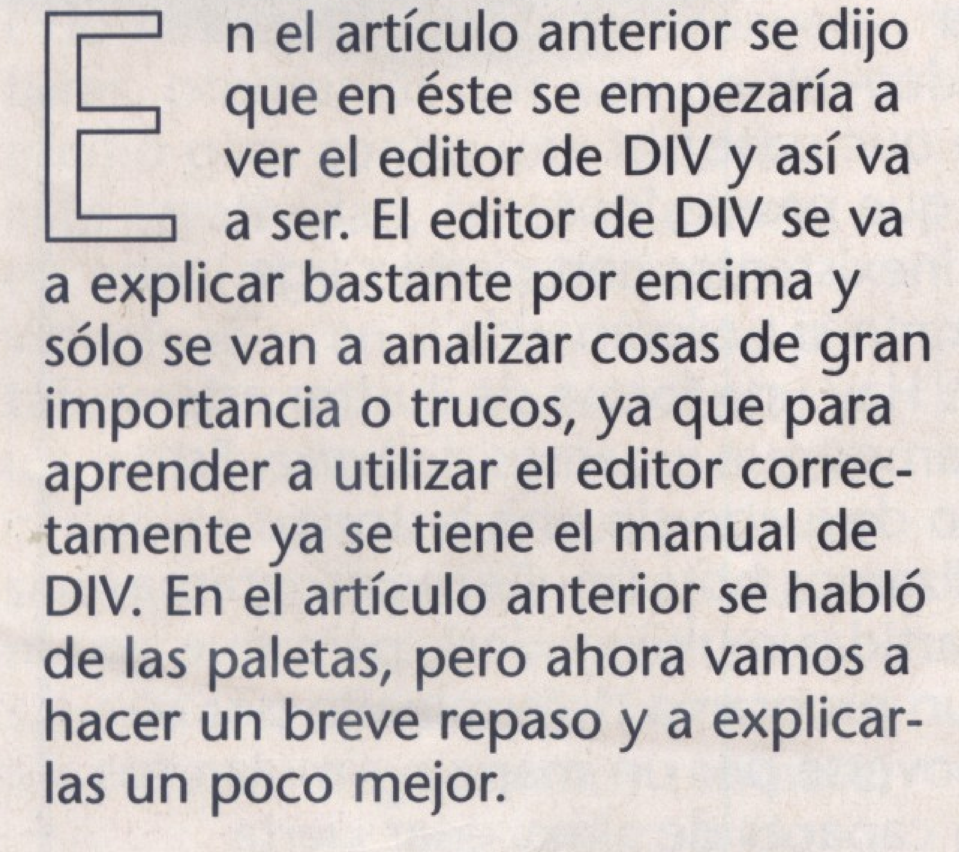
Gonzalo Suárez: Lo que me parece más atractivo de eso es el deseo que tiene la gente de que se convierta en algo real.

Javier Arévalo: Yo no lo considero algo especialmente atractivo. El problema es que en este tipo de juegos el jugador pierde "protagonismo". Ya no es el líder de una tribu o la única esperanza de que la tierra se salve, por poner un par de ejemplos, sino que sus acciones se pueden ver desplazadas por las de otros jugadores.

Gonzalo Suárez: Estoy de acuerdo, pero este hecho es lo que diferenciará a un tipo de juego de otro.

Introducción al diseño gráfico

Primeros pasos



Por fin nos vamos a encontrar con una extensa explicación de las posibilidades que ofrece el entorno de desarrollo DIV con el aspecto gráfico de los juegos. No es un estudio exhaustivo, pero nos proporciona las principales claves para moverse con agilidad con DIV.

En el artículo anterior se dijo que en éste se empezaría a ver el editor de DIV y así va a ser. El editor de DIV se va a explicar bastante por encima y sólo se van a analizar cosas de gran importancia o trucos, ya que para aprender a utilizar el editor correctamente ya se tiene el manual de DIV. En el artículo anterior se habló de las paletas, pero ahora vamos a hacer un breve repaso y a explicarlas un poco mejor.

Las paletas en DIV Games Studio

Una paleta es un fichero que contiene los 256 colores que simultáneamente se van a mostrar en pantalla, o sólo una porción de ellos. Es decir, que todos los colores que se vayan a mostrar simultáneamente en la pantalla de un juego deben estar incluidos en esta paleta. Esto limita bastante las cosas, pero con algunos trucos es fácil acabar consiguiendo lo que se quiere. Imaginemos que tenemos un juego en que primero sale una imagen de presentación y luego empieza el juego. Lo que se puede hacer es cargar una paleta con los colores de la imagen de la presentación, cargar la misma, después hacer un fundido de pantalla y mientras la pantalla está de color negro, volver a hacer un cambio de paleta que se adapte mejor al juego. En el artículo anterior ya explicamos cómo hacer que una imagen creada desde fuera de DIV creara una paleta que aprovechase al máximo todos los colores.

A pesar de las limitaciones de las paletas de valores, es fácil conseguir el efecto que se desea utilizando algunos trucos

NOTA: Si estamos en DIV y queremos empezar a hacer otro FPG (Fichero para gráficos) es recomendable cerrar otros FTP que tengamos abiertos y sus correspondientes gráficos. De lo contrario, los adaptaría a la nueva paleta y en muchos casos eso no conviene.

La fusión de las paletas

Para fusionar varias paletas y conseguir tener una sola que abarque lo mejor de varias, hay que seguir los siguientes pasos. Antes que nada le decimos *cargar paleta* y escogemos una de las paletas que queremos fusionar. Ésta será utilizada por el escritorio de DIV. Una vez hecho esto le diremos *fusionar paletas.* Al hacer esto, DIV nos hará escoger otro archivo. Una vez escogido, el escritorio de DIV utilizará una fusión de las 2 paletas que hemos usado. Seguidamente le diremos guardar paletas y ya estará nuestra nueva paleta guardada. Ahora, cuando carguemos diferentes mapas, al FPG le decimos que queremos que se adapten a la nueva paleta. En la imagen 1 podemos observar un ejemplo del menú de fusionar paletas.

Menú PALETAS
Abrir paleta...
Guardar paleta...
Editar paleta...
Ordenar paleta...
Fusionar paletas...
Mostrar paleta

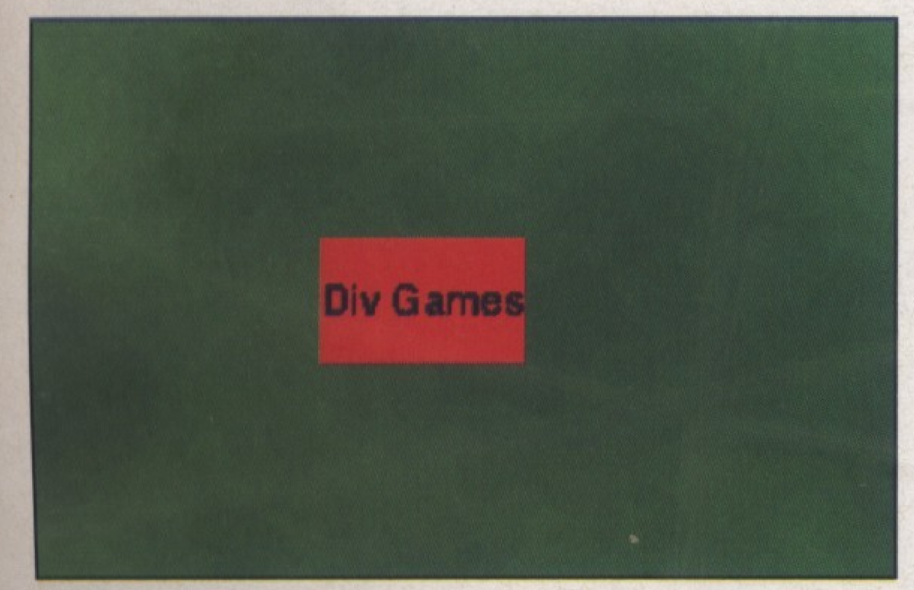
Vista del menú del paletas.

El efecto de transparencia

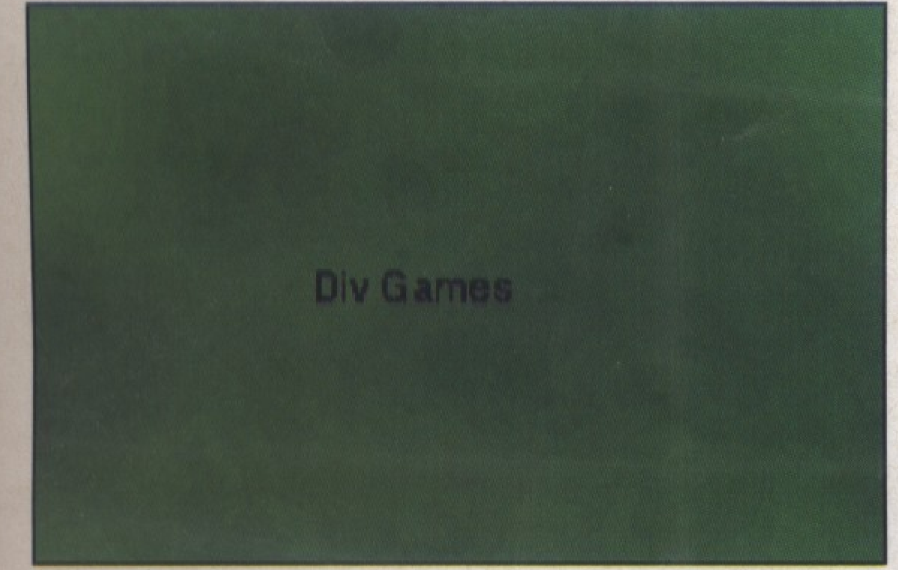
Uno de los efectos más buscados a la hora de realizar juegos es hacer colores transparentes. Como todos sabemos o deberíamos saber, las imágenes son cuadradas o rectangulares pero no pueden tener formas uniformes. Pero si eso es verdad, es difícil hacer un efecto de un objeto que está en un sitio. Por ejemplo, imaginemos que tenemos una imagen de una selva como imagen de fondo en DIV. Seguidamente queremos poner la imagen de un mono delante pero... si no podemos hacer nada transparente, veremos al mono rodeado de una masa de algún color en especial, cosa que provocará que el usuario se sienta incómodo. La forma de solucionar esto es muy sencilla, ya que el primer color de cualquier paleta pertenece al color transparente. Entonces, solucionar nuestro anterior problema sólo se basa en pintar lo que hay alrededor del mono del mismo color que el 1º de la paleta. En la imagen 2 y 3 podemos observar ejemplos de utilización y no utilización de transparencias.

Lo explicado hasta ahora es lo más importante que debemos saber acerca de cómo utilizar el editor gráfico de DIV y las paletas de las





Las letras en imagen sin transparencia



Las letras en imagen con transparencia

imágenes. A partir de aquí, mediante programas de diseño en 2D o 3D, es posible hacer gráficos para nuestros juegos o aplicaciones creadas con DIV GAMES STUDIO. Esto es muy útil a la hora de hacer un juego de plataformas, ya que el personaje puede tener una forma dinámica, y a la hora de detectar colisiones, DIV ignorará los *pixels* de color transparente, cosa que aún le dará más dinamismo al juego.

Construcción de un simple programa

Seguidamente se va a generar un sencillo programa para ver la gran utilidad de hacer colores transparentes, por ejemplo, en los juegos de naves espaciales o bien en los de plataformas. Veremos que, creando un mismo tipo de código para el programa, podemos conseguir distintos efectos gracias a las transparencias.

Para generar este programa sólo utilizaremos un par de gráficos. El autor de este artículo ha utilizado una nave espacial y un meteorito.

Ahora se analizará el código para una mejor comprensión de lo que se hace.

```
program divmania;
begin
    set_mode(m800x600);
    load_fpg("divmania.fpg");
    nave();
    meteor();
end

process nave()
begin
    write(0, 130, 50, 4,
"Ejemplo de la colision con trans-
parencias");
    graph=1;
    x=360;
    y=500;
    loop
if(key(_left))if(x>110)x=x-8;end
        end
if(key(_right))if(x<690)x=x+8;end
        end
        if(key(_up))if(y>100)y=y-
8;end
        end
if(key(_down))if(y<500)y=y+8;end
        end

        frame;
    end
    frame;
end

process meteor()
begin
    graph=2;
    x=560;
    y=170;
    loop
        frame;
    end
end
```

Antes que nada creamos un programa con el nombre *divmania.* Seguidamente cambiamos la resolución de la pantalla a *800x600* y cargamos el fichero para gráficos llamado *divmania.fpg.* Finalmente se carga el proceso *nave*, que se encargará de mostrarnos la nave en pantalla.

En este proceso lo que se hace es, primero, poner un texto que dirá *Ejemplo de colisión con transparencias.* Seguidamente se crea el gráfico de la nave y se coloca en unas coordenadas *X* e *Y.* Después se hace un bucle con las opciones de movimiento de la nave y finalmente se muestra el resultado en pantalla.

> El efecto de transparencia es uno de los más importantes y los buscados a la hora de proponer un juego. Creando un mismo tipo de código para el programa podremos conseguir distintos efectos gracias a las transparencias.

Lo mismo sucede con el proceso *meteor* que nos muestra un pequeño meteorito en la pantalla, pero de momento ninguna opción.

Como podemos observar, con este código lo único que haremos es mover nuestra nave por la pantalla. Ahora se añadirá una línea más para que pueda detectar las colisiones con el meteorito.

Añade la siguiente línea en el proceso nave justo antes de finalizar con el bucle.

```
if(collision(Type
meteor))fade_off();exit("Gracias por
Jugar",0);end
```

Cuadro final

```
program divmania;
begin
    set_mode(m800x600);
    load_fpg("divmania.fpg");
    nave();
    meteor();
end

process nave()
begin
    write(0, 130, 50, 4,
"Ejemplo de la colision con trans-
parencias");
    graph=1;
    x=360;
    y=500;
    loop
if(key(_left))if(x>110)x=x-8;end
        end
if(key(_right))if(x<690)x=x+8;end
        end
        if(key(_up))if(y>100)y=y-
8;end
        end
if(key(_down))if(y<500)y=y+8;end
        end

        frame;
        if(collision(Type
meteor))fade_off();exit("Gracias
por Jugar",0);end
    end
    frame;
end

process meteor()
begin
    graph=2;
    x=560;
    y=170;
    loop
        frame;
    end
```

Como puede observarse, aquí lo único que se hace es mirar si la nave choca con el meteorito y, de ser así, finalizar el programa retornando el texto "GRACIAS POR JUGAR".

Ahora podremos comprobar la eficacia de las transparencias. Prueba a hacer que el primer color de la paleta sea el mismo que el del fondo de la nave. Y, después, pruébalo cambiando el color. Podrás observar que la detección de la colisión es diferente y que este aspecto influye mucho en los juegos o aplicaciones realizadas con *DIV GAMES STUDIO*.

Conceptos básicos de 3D

Cambiando un poco de tema, se mostrarán los conceptos básicos de 3D para aquellos que quieran hacer sus gráficos en 3D. Primero se harán los conceptos básicos porque si empezáramos a explicar el funcionamiento de algún programa 3D, sin estos conceptos no iríamos a ninguna parte.

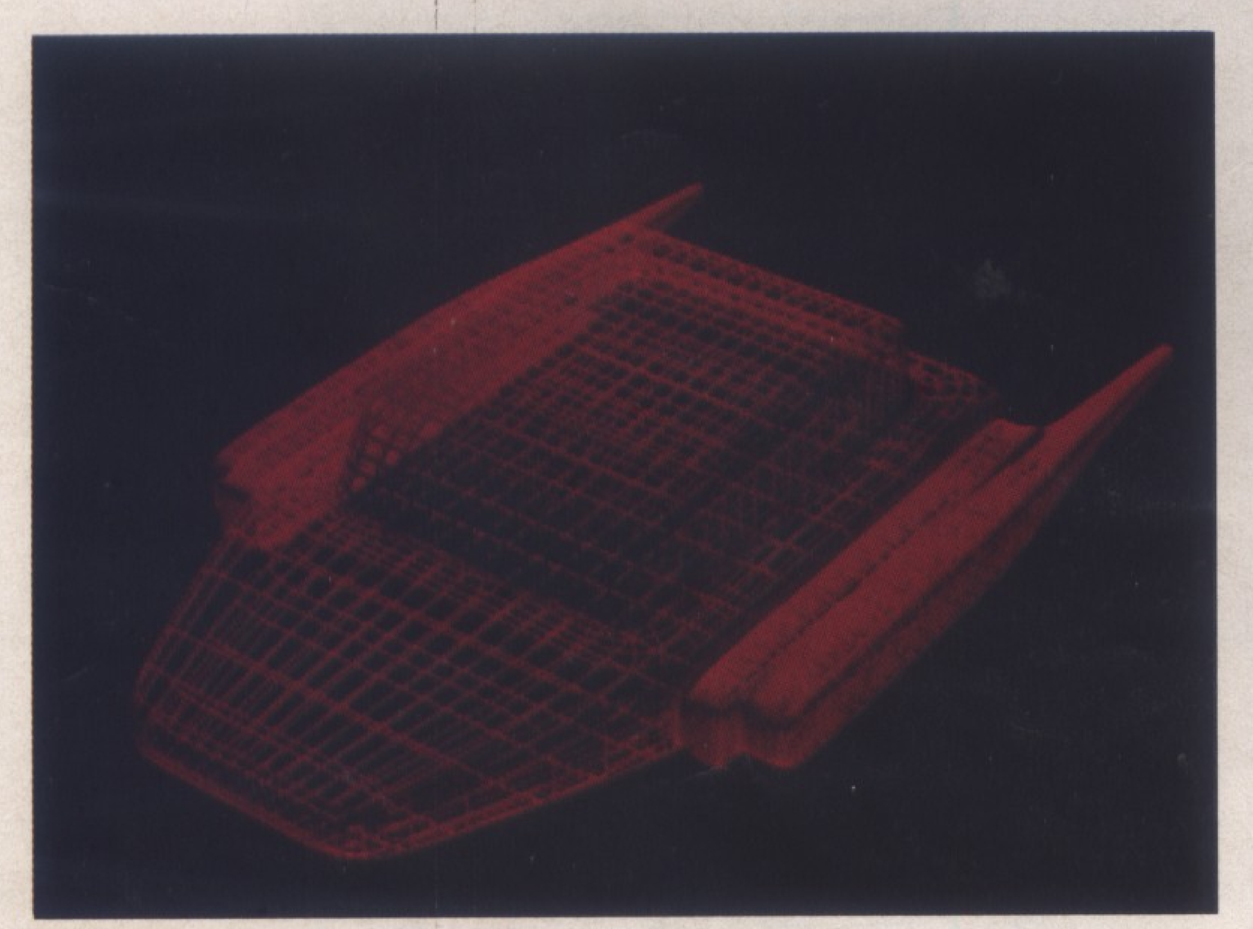
Los conceptos lógicos de 3D son elementales para crear imágenes atractivas. Los pasos que seguir son: el modelado, texturizado, iluminación, animación y el render

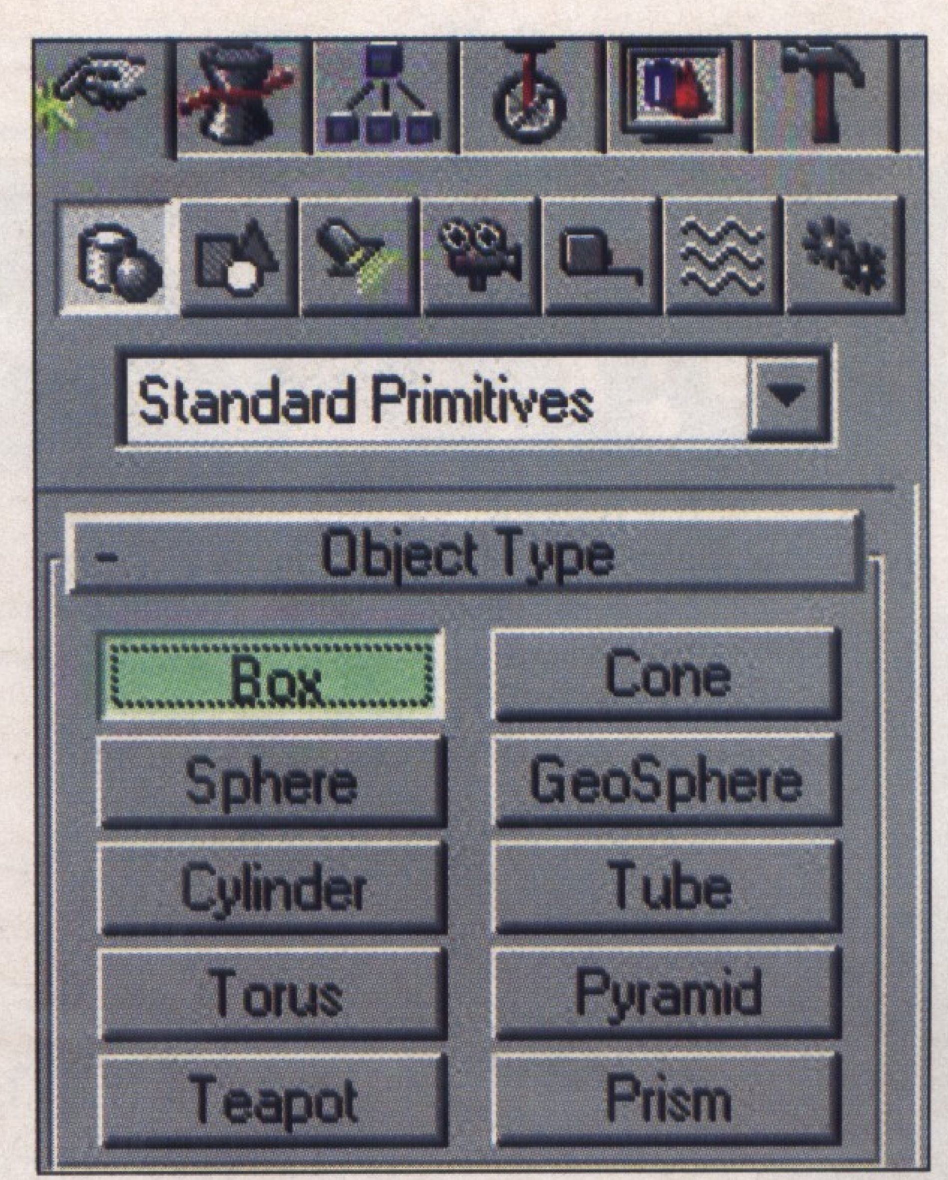
Los conceptos básicos de 3D son imprescindibles para poder crear imágenes impactantes para nuestros juegos o aplicaciones realizadas desde DIV.

Para la creación de escenas 3D los pasos que seguir son: el *modelado*, el *texturizado*, la *iluminación*, la *animación* y finalmente el *render* (representación final de la escena). Los programas de 3D trabajan con objetos formados a partir de vértices, caras y aristas distribuidos en el espacio.

En la imagen 5 podemos observar un simple objeto 3D.

Para modelar estos objetos hay que recurrir a las herramientas de modelado del propio programa. Con estas herramientas podemos modelar desde simples primitivas, como cubos o esferas, hasta complejas naves espaciales o realistas dinosaurios.


Un simple objeto 3D.



Creación de un cubo.

Hay que decir que también existe software que sólo sirve para modelar y luego exportar el objeto a otros programas, como en el caso de Rhinoceros 3D.

El siguiente paso en la creación de escenas es texturizar los objetos. Este paso es muy importante, ya que un objeto muy bien modelado pero mal texturizado acaba pareciendo falso. Las texturas forman parte de los materiales, que son a su vez las propiedades de brillo, rugosidad, color, transparencia...

Todos estos parámetros se pueden definir según las texturas (*bitmaps*). Veamos un ejemplo de ello.

Crearemos un cubo en tres dimensiones y le aplicaremos un material con 2 texturas. Una definirá el color y la otra definirá la rugosidad. Para esto vamos a utilizar 3D STUDIO MAX, pero los conceptos son extrapolables a cualquier programa 3D.

Antes que nada crearemos el cubo. Para ello vamos a la pestaña de creación, hacemos clic en *primitivas* y después en la opción *cubo*.

En la vista SUPERIOR arrastramos el ratón con el botón apretado, creando así un cubo en 2D. Luego soltamos el botón y arrastramos hacia arriba para darle la altura. Debemos asegurarnos de que la opción *GENERAR MAPAS DE COORDENADAS* está activada. De lo contrario, la textura no se vería bien en el resultado final.

Ahora hacemos clic en el icono de editor de materiales para definir el material (el icono del editor de materiales es uno que tiene unas bolas de colores).

Al activar el editor de materiales ya tenemos un material activado, por lo que editaremos sus mapas pulsando en el botón *DIFUSO* de la persiana *MAPAS*. Aparece ahora una ventana con una lista de los tipos de mapa disponibles. Escogemos *BITMAP*, ya que se trata de una imagen en 2D. Seguidamente nos aparece un cuadro de opciones y un botón para seleccionar la imagen que servirá para darle color. Picamos en el botón y la buscamos en el disco duro. Para volver a la persiana mapas pulsamos el icono como el de la figura que tiene una flecha dibujada. Este icono esta también en el editor de materiales.

Ahora repetimos los mismos pasos con la opción de RUGOSIDAD. Esto sirve para que el material parezca tener relieve según el valor en escala de grises de los *pixels* de la imagen escogida. Este mapa viene por defecto con un valor de 30 %, pero se puede aumentar para conseguir un efecto de mayor rugosidad (también le afectan números negativos).

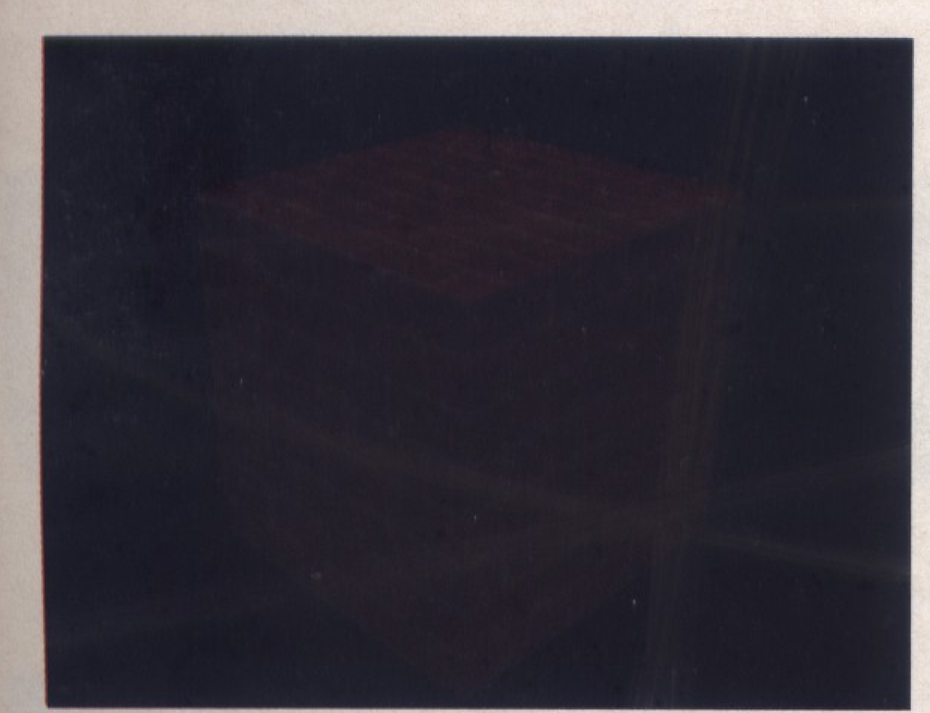
Una vez definido el material, lo aplicamos pulsando en el icono *APLICAR MATERIAL* (con el cubo seleccionado). Seguidamente podemos ver dos imágenes, la imagen 6 y la 7 con los resultados del cubo con rugosidad y el cubo sin rugosidad.

La iluminación

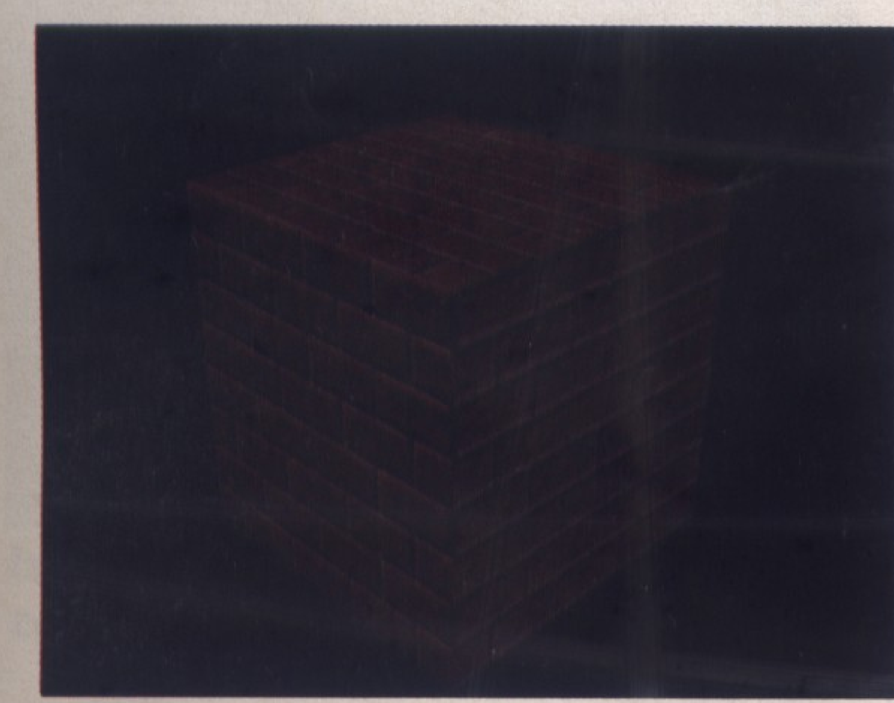
La iluminación es un factor muy importante, ya que se puede conseguir imágenes muy reales trabajando un poco las luces. La mayoría de paquetes actuales crea una iluminación por defecto, aunque es muy simple. Casi todos los programas soportan varios tipos de luces, como pueden ser luces omnidireccionales (como una bombilla, o sea, radiales) de foco (¿necesita explicación?), paralelas (como la luz solar), etc. Hay programas que soportan también efectos de radiosidad, que dan un realismo increíble a la escena, aunque el cálculo que esto conlleva es enorme: puede tardar incluso días una escena bastante simple. La radiosidad es la iluminación causada por el rebote de los rayos de luz en los objetos. Aunque muchos programas no lo soporten, el efecto de radiosidad se puede simular con una buena iluminación y mucha paciencia.

El *render*

Llamamos RENDER al procesado final de la imagen. Cada programa



Cubo sin rugosidad.



Cubo con rugosidad.

tiene su propio motor de *render*; dependiendo de él se pueden conseguir diferentes resultados. El *render* es un proceso lento, y cuanta más información tenga nuestra escena, más lento será el *render*. El resultado del *render* es una imagen o secuencia de imágenes que podemos guardar en nuestro disco y que podemos abrir y retocar con programas de retoque. Cada motor de *render* tiene sus propios parámetros de configuración, por lo que sería imposible hablar de todos ellos en este espacio.

Conceptos básicos de la animación

La animación es una de las tareas más complejas a la hora de generar una escena. Su dificultad se basa en que si queremos representar cosas reales tenemos que hacer un movimiento realístico. Por ejemplo, todos sabemos cómo camina una persona; sin embargo, hacerlas caminar es un proceso muy complejo.

Para la animación de personajes existen varios paquetes, como Character Studio (*plugin* de 3DS MAX de Kinetix). Con él es posible hacer animaciones bastante realistas, pero como el movimiento de nuestro cuerpo no siempre es igual y es muy complejo, este *plugin* también lo es. No vamos a explicarlo aquí porque no tendríamos ni espacio para introducirnos al *plugin*.

No sólo la animación de personajes es compleja, sino que la animación de cualquier cosa es muy difícil... Por ejemplo, el movimiento de una flor al entrar en contacto con el viento es bastante complejo, aunque no lo parezca.

Algo que ayuda mucho a la animación y que le dará más vida son efectos como los de *BLUR* que difuminan la escena o un objeto que está en movimiento para que no se vea siempre tan nítidamente.

También son importantes en la animación el movimiento de anticipación, las posiciones de esfuerzo y, finalmente, el movimiento que se va a hacer. Es decir, cuando una persona salta, por ejemplo, no va caminando, sale volando hacia arriba y después cae; antes de saltar, tiene un movimiento de anticipación en el que flexiona las piernas y estira los brazos y después pega el bote flexionando las piernas y brazos. Lo mismo sucede con la caída, en la que después de poner los pies en el suelo, la persona en cuestión vuelve a flexionar las piernas, aunque no tanto como antes de saltar, y después se pone en la posición inicial.

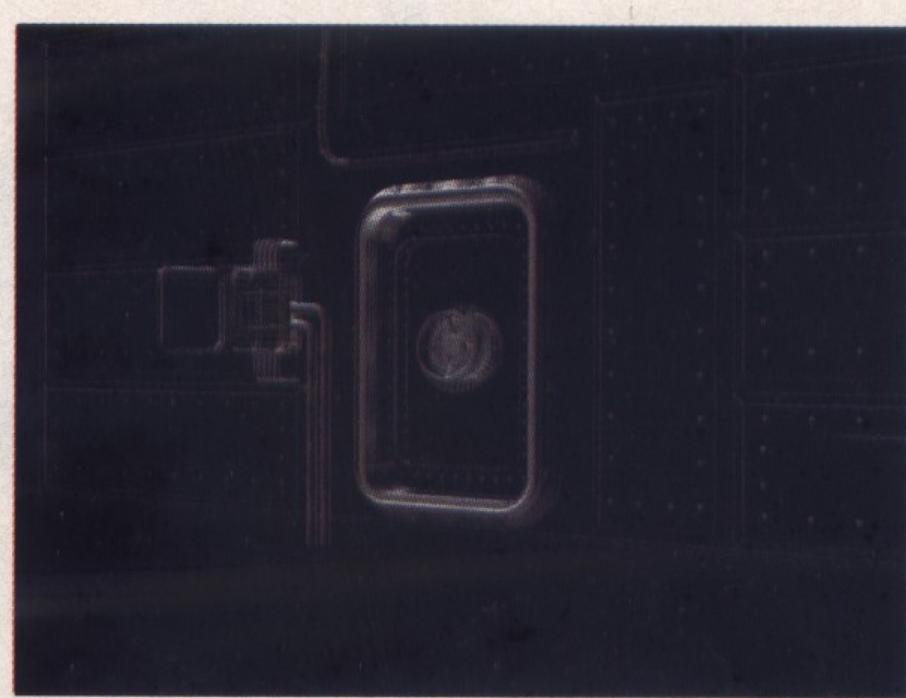
Esto puede parecer lógico pero no será que nunca habéis visto un videojuego con tales fallos. Al ver el resultado final, parece un movimiento real y pasamos por alto que hay fallos, pero si miramos una animación o un juego en el que esto no pasa nos daremos cuenta en seguida de que algo le falla al personaje y si investigamos un poco podremos observar que esto es debido al movimiento.

Recomendaciones

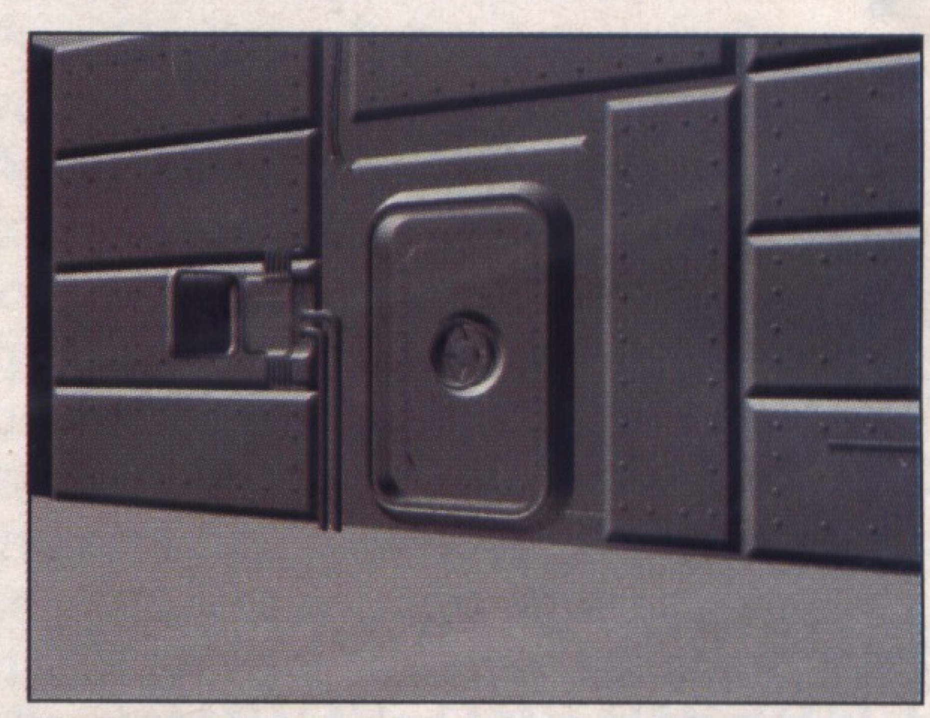
La utilización de colores croma (colores que actuarán como transparentes). Es decir, a la hora de hacer un gráfico, hacerlo con un fondo verde fluorescente o azul, de manera que el negro se pueda utilizar en el juego sin miedo a que, por ejemplo, nos salga una persona con los ojos transparentes.

Limpiar las zonas transparentes. Si usamos un escáner o hacemos un dibujo directamente, a veces quedan regiones del gráfico con unos píxeles inútiles. Esto es un peligro a la hora de hacer transparencias, ya que las colisiones se provocarían también al encontrar este píxel, cosa que no se quiere. Cuando vayas a utilizar colores croma (explicado más arriba), ten en cuenta este detalle.

Hacer unas composiciones agradables a la vista y fáciles de ver. Como es lógico, a la hora de hacer gráficos se debe pensar en los colores que se van a poner y el efecto



"Imagen 8 – Modelado".



"Imagen 9 – Modelado sin texturizar".



"Imagen 10 – Texturizado pero sin iluminación trabajado".

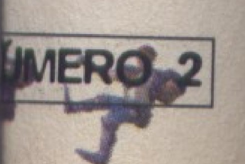


"Imagen 11 – Escena completa".

que darán sobre el usuario de nuestro programa. Es decir, si nuestro programa tuviera que inspirar tranquilidad, utilizaríamos colores cálidos, mientras que si queremos expresar nerviosismo o miedo deberíamos utilizar colores oscuros y vivos a la vez, como el negro y el rojo.

Hacer gráficos sencillos pero expresivos. Es importante "currarse" los gráficos de un juego o aplicación, pero se debe tener cuidado de no hacerlos demasiado complejos porque podrían acabar agobiando al usuario.

Por David Martínez (*d_martinez@geocities.com*) y Alex Huguet



Island Dream: el sueño de u

Compañías con un gran futuro

Allá por 1989 en Palma de Mallorca se creó uno de los mejores grupo de desarrollo independientes de España. Island Dream, con cerca de una veintena de títulos en el mercado, se ha consagrado como una leyenda; tocando arquitecturas tan dispares como PC, recreativas, Amiga y tragaperras digitales, Es sin duda un ejemplo a seguir por cualquier grupo freelance que se precie.



Compañía:	Island Dream
Juego en desarrollo:	Necro
Género:	Acción
Título previo:	Mr Tiny Adventures.

Mr. Tiny Adventures es el último proyecto que Island Dream logra terminar a lo largo de 10 años de producción, con el que obtuvieron el primer puesto en el Concurso Nacional de Videojuegos de la revista PC ACTUAL en 1998. Island Dream ya había ganado el primer puesto en la 2ª edición de este concurso hace unos cuantos años con la videoaventura *ELFO*, que cristalizaría poco después en un juego de dos partes publicadas en la revista Computer Gaming World, *La luz del Druida* y *Arakhas el Oscuro*. Recientemente ha recibido el primer premio en el concurso de Stratos; el «stratino de oro» al grupo con mejor trayectoria. Una merecida mención a su brillante evolución. Con once videojuegos para PC, dos máquinas recreativas, tres tragaperras y un título en Amiga, el comentario de toda su producción nos llevaría varias páginas. Seguro que recordáis títulos como *Steel Force, Mortal Race, Chaos Maze, Hot Speed, Deep Red, Eco Kid* y los más recientes *La Luz del Druida, Arakhas el Oscuro*, y sin duda sus mejores títulos hasta el momento: *Galax Empires* y *Mr. Tiny Adventures*. Podéis pasar por su web en www.islanddream.com para más información. Con esta trayectoria, no se nos podía escapar la oportunidad de tener una entrevista exclusiva con Island Dream...

¿Quiénes fuisteis los fundadores de Island Dream y quiénes componen actualmente el grupo?

El grupo fue fundado en el año 1989 por Javier Maura (programador) y Miguel A. Carrillo (grafista) y nació para la creación de videojuegos para ordenadores Amiga, que en aquella época ofrecían unas características técnicas muy avanzadas. Mucho tiempo ha pasado desde entonces y durante el mismo hemos tenido muchos colaboradores, aunque el núcleo principal siempre ha sido el formado por Javier y Miguel A. hasta el año pasado, en el que Javier crea un grupo independiente ("siempre quise ser un elfo"), y se incorporan a Island Dream dos nuevos programadores, Toni Pomar y Juanga Covas, que junto a Miguel A. continúan con el proyecto.

¿Con qué proyectos vais a sorprendernos a corto y medio plazo?

Probablemente, cuando salgan estas líneas a la calle podremos encontrar en el mercado el videojuego *Mr. Tiny*, la última producción del grupo. Para dentro de unos meses se podrá ver una *alpha* del nuevo proyecto *Necro*, videojuego de acción donde el equipo de grafistas está poniendo todo su empeño. El programa está ambientado en un futuro próximo post-apocalíptico donde deberemos sobrevivir a los más insospechados peligros.

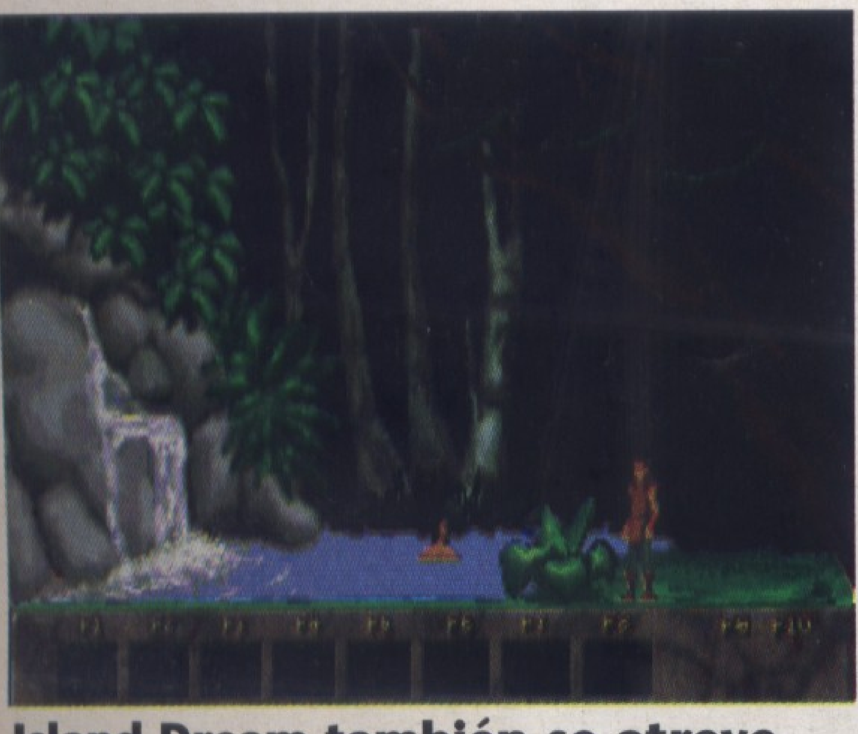
A más largo plazo, y como gran proyecto, se espera el lanzamiento de *Elha*, una aventura épica en un inmenso continente lleno de magia, misterio y acción, donde nuestro protagonista podrá recorrer a sus anchas el mapeado y se encontrará con las más variadas aventuras y desventuras.

¿Qué utilizasteis para elaborar los impresionantes gráficos y el fantástico *engine* del *Mr Tiny*?

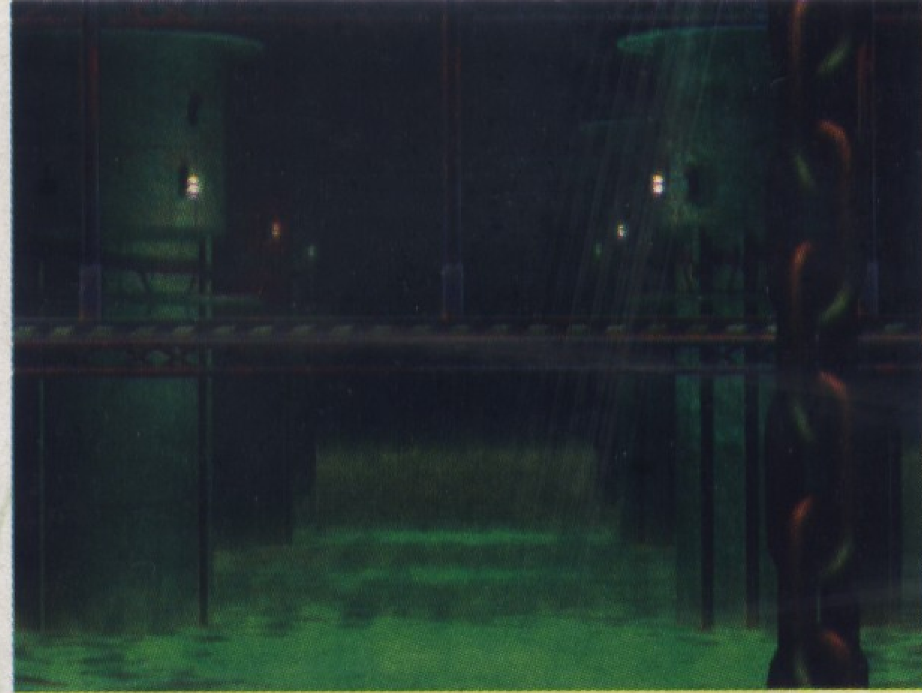
Los gráficos han tenido dos tratamientos muy diferentes uno de otro. Para la realización de los fondos y de las fases se han empleado gráficos renderizados con Lightwave 3D, procurándo darles un aspecto realista, pero colorido a la vez.

Los personajes, enemigos, items, etc., se han realizado utilizando técnicas tradicionales de dibujo y animación 2D lo que, unido a los fondos 3D, confiere al juego un aire muy especial. Para la creación de las fases hemos contado con un potente editor diseñado a medida para la realización de este tipo de juegos.

de cualquier grupo freelance



Island Dream también se atreve con la estrategia.



Los escenarios no tienen nada que envidiar a los de las grandes casas.

En cuanto a la programación, decir que se ha utilizado el Visual C++ 5.0, bajo Direct X 5, porque eran las tecnologías punteras en el momento de comenzar el proyecto. Nuestro objetivo era conseguir inyectar un nuevo aire al concepto de las plataformas para PC; es decir, más colores, más resolución, más velocidad, etc... En resumen, actualizar tecnológicamente el concepto de los juegos 2D, que se habían estancado en una tecnología obsoleta (256 colores, baja resolución, etc.).

¿Cuánto tiempo medio le dedicáis a cada proyecto?

¡Ésa es la pregunta del millón! Por regla general se tarda más de lo que se pensaba al principio. Todo depende de lo que se pretende hacer; cada proyecto es un mundo diferente pero, como norma, un juego de tipo bajo-medio no debería durar más de 4-8 meses. Si lo que se pretende es hacer un gran proyecto, el tiempo se puede disparar: un año, dos...

A veces es difícil cuantificar el tiempo real de desarrollo, en nuestro caso, y como trabajamos como freelances, no podemos dedicar todo el tiempo que quisiéramos, pues cada uno de nosotros tiene que acudir a sus trabajos habituales y dedicar parte de su tiempo libre al desarrollo. Por eso nos solemos tomar las cosas con más calma. De todas formas, al final de cada proyecto siempre vamos con prisas, sin descansar ni en los fines de semanas y durmiendo muy poco.



Mr Tiny cuenta con unos gráficos impresionantes

¿Qué consejos podéis darle a los grupos independientes que comienzan ahora en el mundillo del desarrollo de videojuegos?

El primordial consejo que dar a los grupos que empiezan o se quieren dedicar a esto es el siguiente: Constancia. Uno de los males que puede tener un proyecto es que poco a poco va mermando el interés de los desarrolladores. Es decir, que nos va aburriendo a medida que se alarga el periodo de creación. La única manera de seguir adelante es tener la suficiente constancia y saber motivar a los compañeros para continuar.

Es importante saber también la opinión que tiene del proyecto alguna persona ajena al mismo, pero cualificada para su evaluación. Desde la creación de la asociación de desarrolladores Stratos, esa labor es fácilmente encomendable a sus coordinadores, que gustosos orientarán al grupo sobre los posibles defectos o mejoras posibles, además de poder empezar a mover el producto de forma comercial, mediante demos, etc. Todo esto ayuda a que el interés se siga manteniendo y a crear posibles expectativas económicas... ¡que nunca vienen mal!

¿Creéis que nos encontramos de nuevo con un «renacimiento» del software español como ocurrió hace años en la época de Spectrum, MSX, etc.?

Evidentemente, el mundo del desarrollo de software en España se está moviendo a una velocidad infernal. Estamos ante un momento ideal para el sector. Grupos como Pyro Studios o Rebel Act están llevando el nombre del software español al más alto nivel internacional y todo ello repercute directamente en los beneficios de las empresas, lo que las motiva para crear nuevos proyectos. La rentabilidad económica está casi asegurada. Un factor importante es que hoy en día ya no se considera extraño tener un ordenador en casa. Además, se une a ello el hecho de que ahora la inmensa mayoría son PCs, con lo cual el mercado se estandariza y facilita la creación de software. Ya no hay que crear varias conversiones para llegar a más gente, como ocurría hace unos años.

Esta pregunta es casi obligada... ¿Qué opináis de la herramienta DIV?

A menudo me encuentro con gente que quiere empezar a programar videojuegos y meterse en el mundillo y me pregunta qué es lo que deben hacer, por dónde empezar etc. Desde que salió al mercado la herramienta DIV, ya no tengo dudas de cómo contestarles. Con esto no quiero decir que DIV sólo sirva para principiantes, ni mucho menos. Estamos ante una herramienta profesional para el desarrollo y la mejor opción a la hora de plantearse un proyecto. A la vez se trata de un sistema sencillo y dedicado, con lo que el aprendizaje se hace muy intuitivo y entretenido. Con la nueva versión de DIV se prevé una serie de mejoras que seguro auparán más arriba las posibilidades de desarrollo. De hecho ya se están haciendo juegos de calidad profesional y que pronto podremos ver en la calle.

Muchas gracias por vuestro tiempo, y suerte con vuestros futuros proyectos.

Queremos desde aquí dar un saludo a todos los grupos independientes y animar a todos los que empiezan en este apasionante mundillo del desarrollo de videojuegos. Entre todos formamos la cantera de lo que será el futuro profesional en nuestro país.

Carlos Glez. Morcillo
(cgonmor@jet.es)

La hora de DIV 2

Las webs reclaman la segunda parte de DIV

Las páginas de Internet dedicadas a DIV Games Studio han crecido, tanto en número como en calidad. Así, el pequeño mundo de la programación de videojuegos centrado en este entorno se ha mantenido despierto y alerta, a la espera de la aparición de la segunda edición de la popular herramienta.



Impaciencia ante el retraso de DIV 2, eso es lo que más llamaba la atención de las páginas dedicadas a este entorno dentro de la Red de redes. También hay algún que otro comentario acerca del retraso de *DIV Manía 2*, pero al final hemos aparecido para acallar todos los rumores, y a estas alturas también habrá hecho su aparición (mucho más esperada que la nuestra) DIV 2, de modo que los autores de estas páginas verán satisfechas sus demandas.

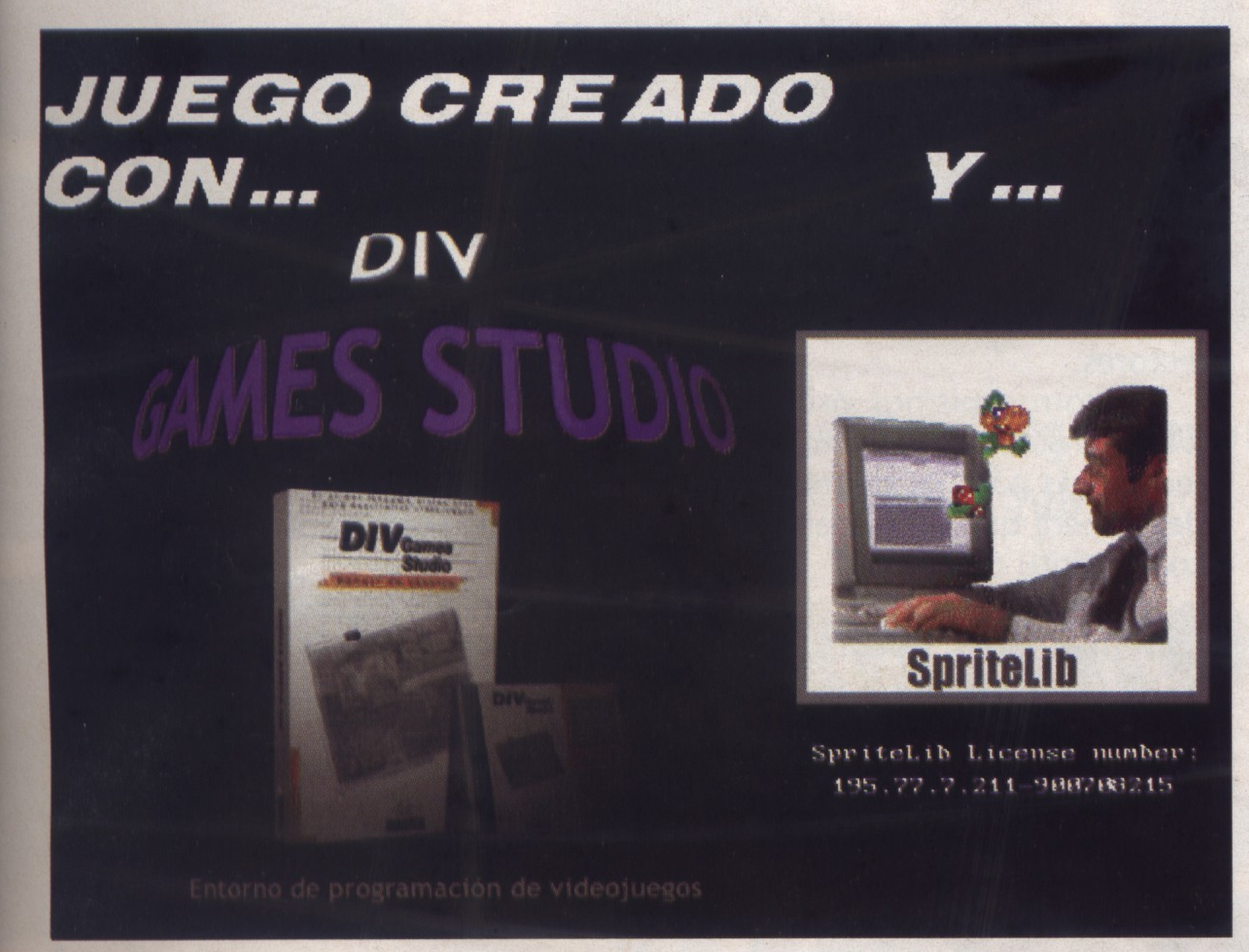
Por lo demás, sigue habiendo un buen número de anunciantes que buscan compañeros para crear algún grupo de programación, basado tanto en DIV como otras herramientas de trabajo, desde Visual Basic hasta Pascal. Todo ello indica que, lejos de mermar con el paso del tiempo, el fenómeno DIV se va afianzando y enriqueciendo con nuevas iniciativas, como la del *FreeDIV*, que ya veremos cómo culmina.

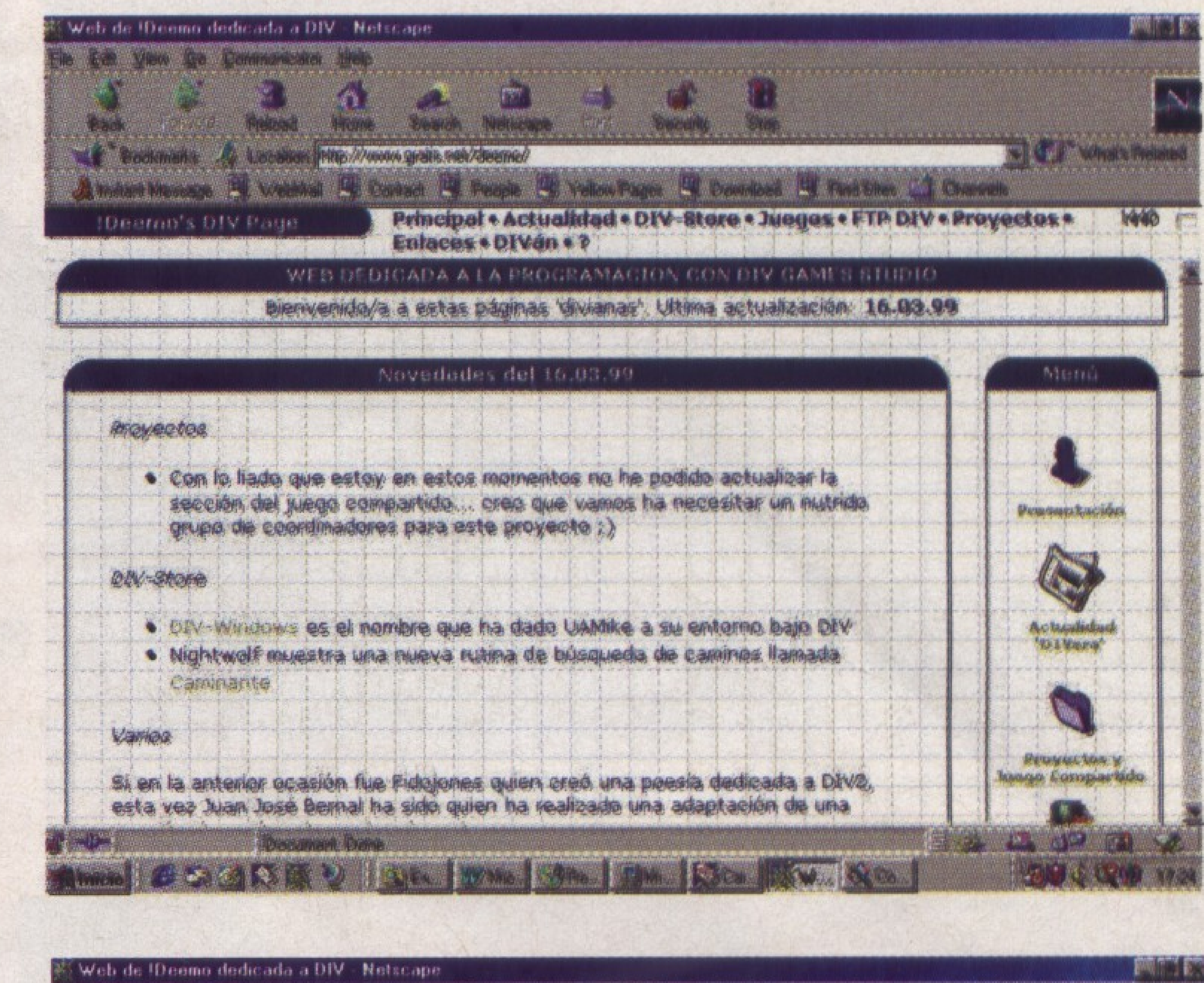


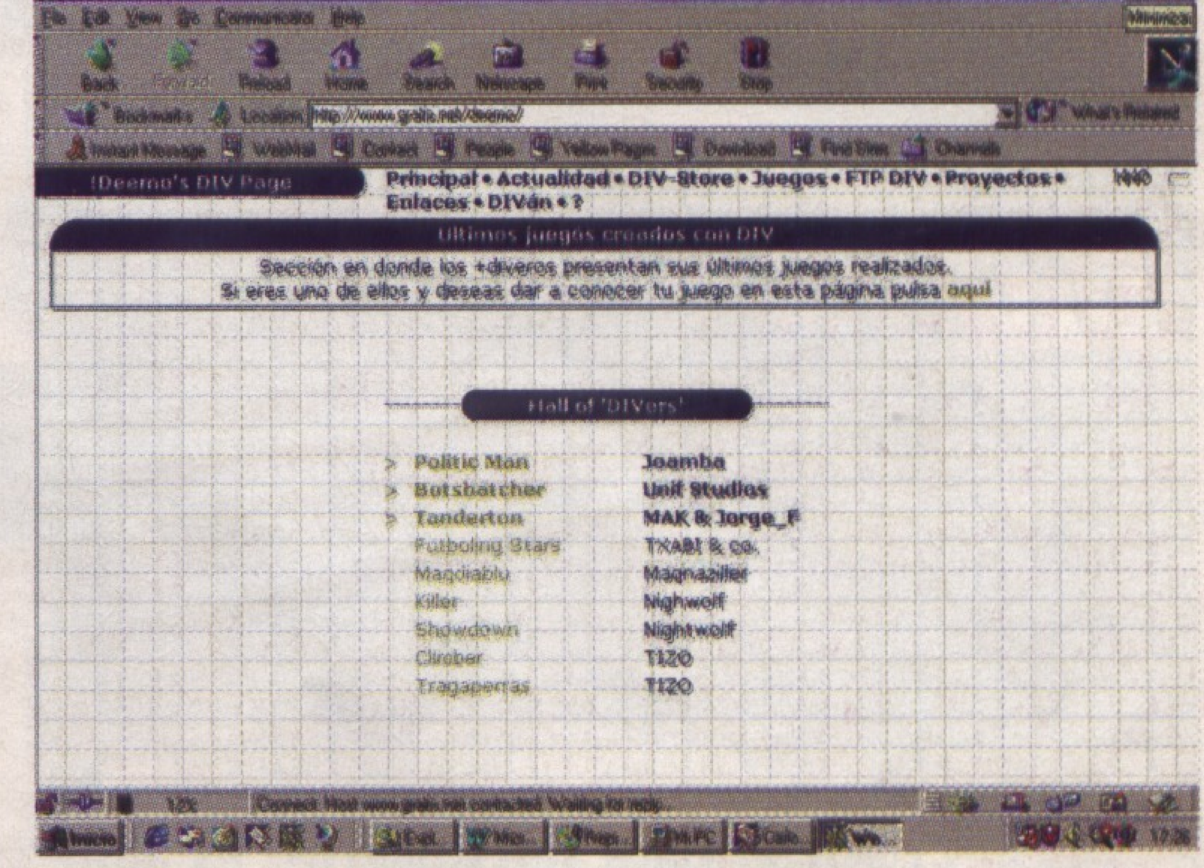
Página oficial

Se ha visto renovada en buena medida, no sólo por el *look* que nos ofrece sino por las cosas que incluye. Ahora veremos unos fondos más claros y diáfanos, y una interfaz de sencillo manejo. Una de las novedades, que más nos puede interesar a nosotros, es que incluyen un apartado dedicado a nuestra revista, con lo que los lectores que todavía no han comprado nuestro último número pueden echar un vistazo para ver si les interesa. No sabemos cuál es la frecuencia de actualización de la página, pero confiamos en que incluyan nuestros contenidos cuanto antes. En una de las imágenes que acompañan este artículo podréis ver la página que dedicaron a nuestro primer número.

Por lo demás, incluye secciones tales como Soluciones, Desarrollo, Programación, Productos, Noticias, etc. Todas ellas están realizadas con el objetivo de abarcar todos los temas concernientes a DIV, desde cómo adquirirlo hasta los grupos de programación que lo utilizan. No faltan secciones como *download*, aunque en estos momentos la que más nos interesa es la dedicada a la nueva edición de DIV, que pueden consultar todos los interesados (aunque,







desde luego, la información que damos en la revista es mucho más completa y actual...).

Tampoco hay que olvidar la página de *links* que, si bien no es tan completa como la de otros amantes de DIV, lo cierto es que nos lleva a alguno de los imprescindibles *sites* dentro de este entorno: lo que nos ofrecen tres de los principales promotores del mismo, Ximo, Deemo y Tizo. Dentro de las páginas de estos aficionados o profesionales sí que podemos encontrar una buen número de enlaces, especialmente en la Ximo. En efecto, este loco de DIV nos ofrece una de las mejores páginas de *links* que es posible encontrar, de modo que quien quiera descubrir este pequeño mundo no tiene más que acercarse a ella y comprobar todo lo que nos puede ofrecer.

Otras aportaciones

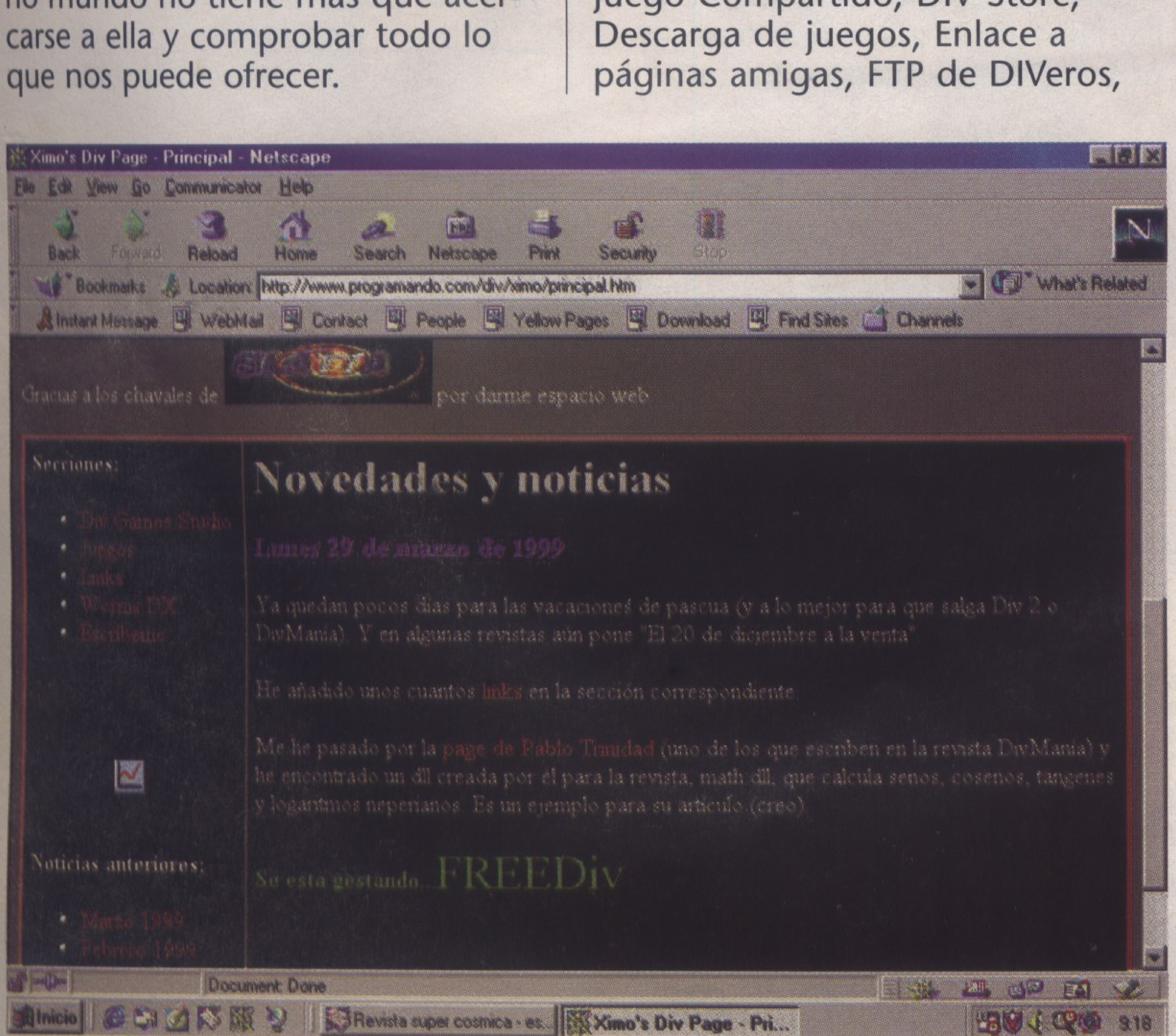
Pero, a pesar de la sensible mejora de la página oficial de DIV, lo cierto es que lo más interesante del mundillo se sigue moviendo en las webs dispuestas por los aficionados, algunos de los cuales se han tomado tan en serio sus trabajos con DIV que empezamos a dudar si son propiamente aficionados o ya han entrado en el campo de lo profesional.

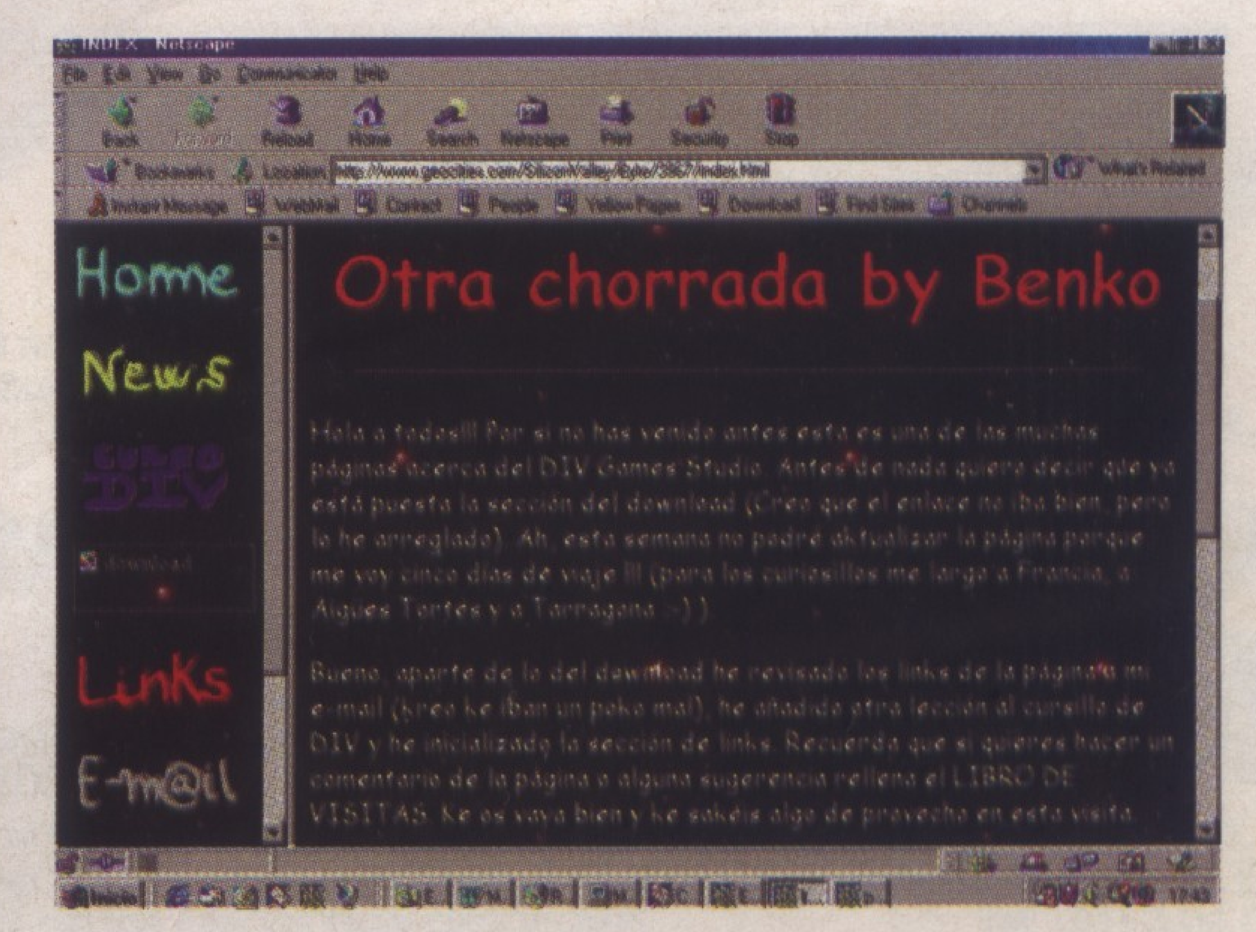
Pues bien, entre todas ellas la que más destaca es la de Deemo, que ya en el número anterior fue señalada como una de las *webs* que mayor interés podían despertar en alguien interesado en el tema. Con secciones como Actualidad DIVera, Proyectos y Juego Compartido, DIV Store, Descarga de juegos, Enlace a páginas amigas, FTP de DIVeros, El DIVan, Webeando y un historial de actualización, se perfila como una de las más completas páginas del momento. Aunque mantiene algunas secciones todavía en construcción, o incompletas, en el momento en que sus posibilidades estén al cien por cien podrá dar mucho de sí.

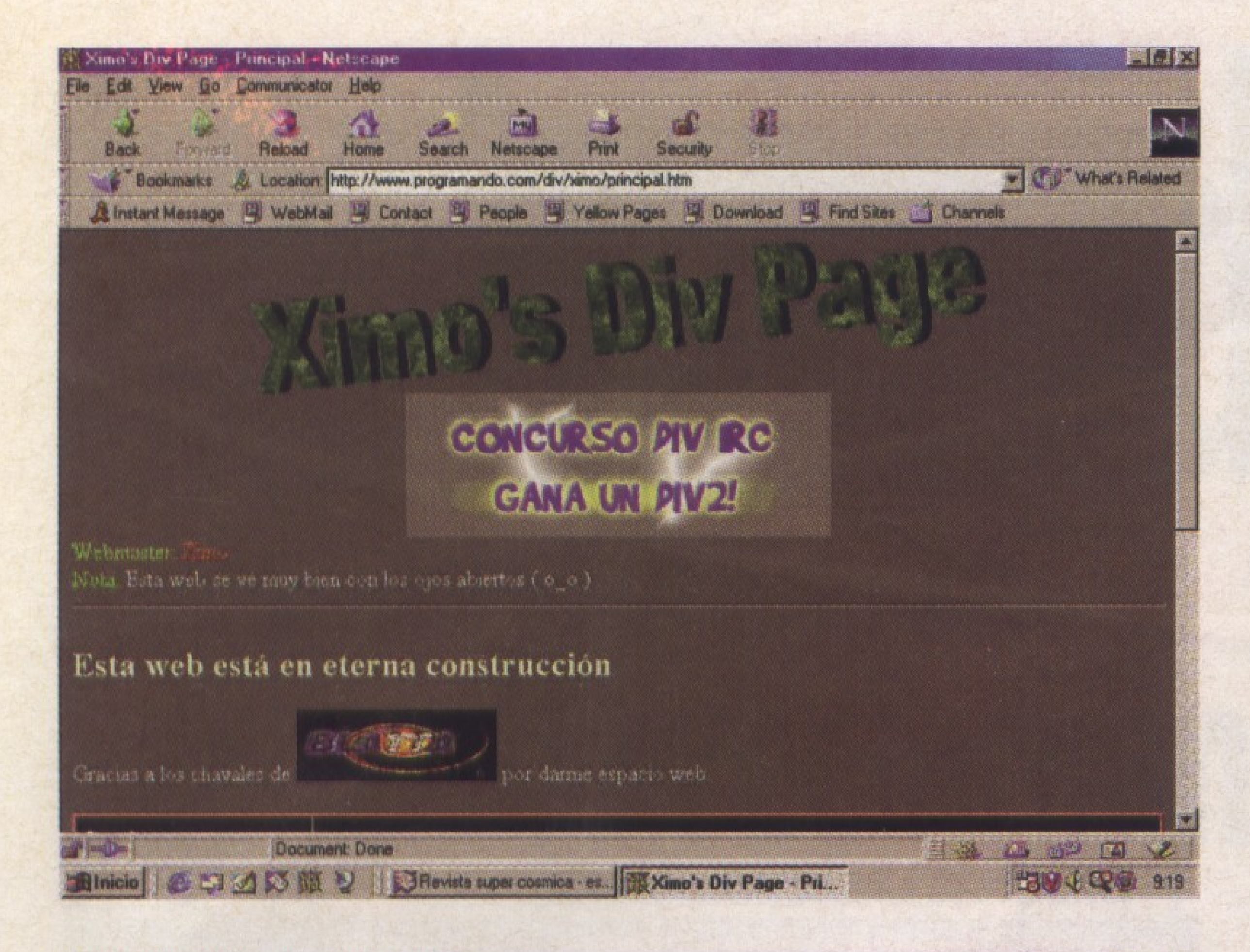
El caso de las páginas que nos ofrece Tizo es distinto. Su apa-

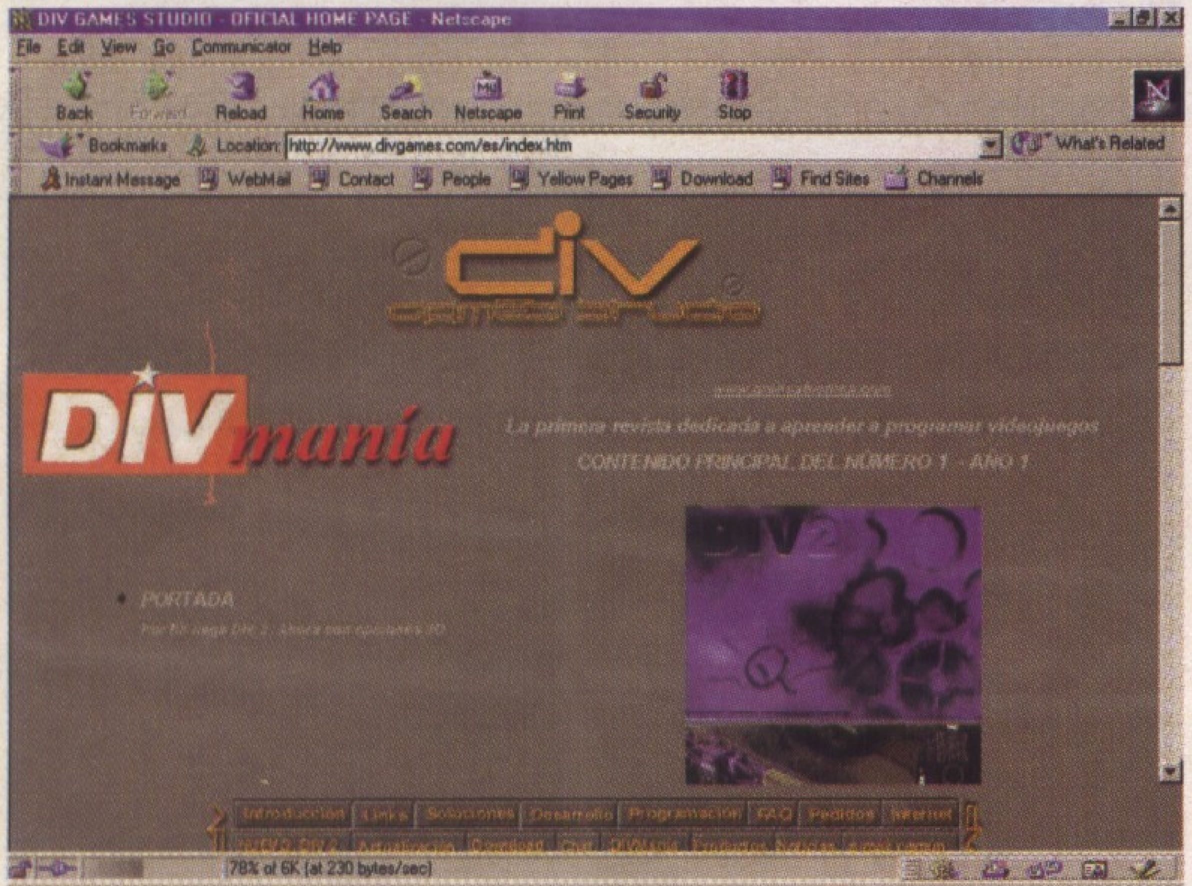
Las 10 mejores DIV Webs

www.divgames.com
pagina.de/tizo
pagina.de/ximo
pagina.de/deemo
www.gratis.net/div/MAQNAZILLER
www.geocities.com/SiliconValley/Byte/3967
www.geocities.com/TimesSquare/Galaxy/5141
personal1.iddeo.es/ret0021o
ctv.es/USERS/emart/divchanel
pagina.de/coldd









Las mejores páginas DIV

Como ya sabéis todos a estas alturas de la revista, DIV Manía desea contar con la máxima colaboración de los lectores. Y especialmente con la de aquellos que tienen una vida activa dentro de la Red de redes. Así, en cada edición de la revista vamos a estar muy atentos a vuestras evoluciones dentro de Internet. Aquellos que tengan páginas web dedicadas a DIV Games Studio no deben dudar en darlas a conocer a la revista, para que desde aquí las mostremos a los demás lectores.

Nosotros mismos visitaremos vuestras páginas web y navegaremos a través de ellas para ver qué ofrecéis a los internautas. Cada vez aparecen más páginas dedicadas a DIV, y su vistosidad es cada vez mayor, de manera que no debéis dormiros en los laureles; seguid adelante con espíritu creativo.

La manera de ponerse en contacto con nosotros es muy sencilla. En primer lugar, puedes darlo a conocer en el canal de chat dedicado a DIV (se llama DIV y se encuentra en arrakis) o bien en la página web oficial del entorno, *www.divgames.com*. También puedes mandar un E-mail a nuestra revista (*divmania@prensatecnica.com*) en el que nos presentes tu página web. Por último, puedes comunicarlo por correo ordinario, a la dirección de la revista, indicando en el sobre: "DIV Manía-Páginas WEB". Nuestra dirección es:

C/Alfonso Gómez 42, nave 1-1-2
28037, Madrid, España

riencia, para ser sinceros, es algo más precaria, y parece que en los últimos tiempos han sido un poco dejadas. Sin embargo, sus contenidos son de los mejores, especialmente en o que se refiere a cursos o a profundización en algunos temas. No en vano, es uno de los programadores que más íntimamente relacionado está con este entorno de desarrollo de videojuegos. Lo conoce a fondo y no tiene empacho en mostrar a quien quiera que se acerque a él todo lo que puede ofrecer.

Programadores

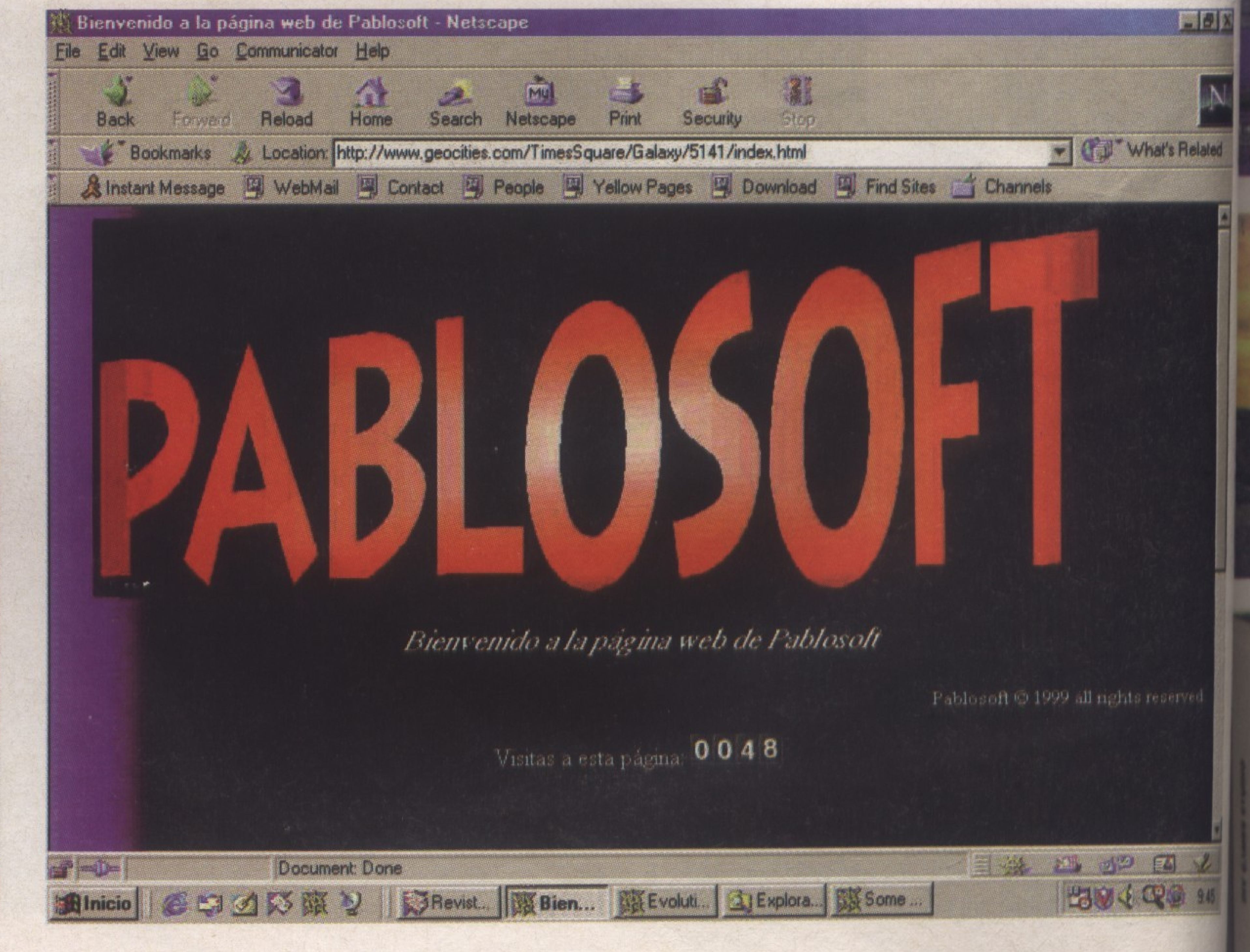
Las más abundantes son las páginas lanzadas por los aficionados a la programación que se interesan en DIV y se dedican a divulgar sus posibilidades y lo que hacen con él. Entre todas las páginas, destacan por su originalidad la de Benko, por sus contenidos la de PabloSoft y por el interés de la versión de *Diablo* la de Maqnaziller. En general, lo más destacado de todas ellas es su sección de *download*, en la que los visitantes pueden bajarse de forma totalmente gratuita los juegos que han programado bajo DIV. Algunos son de calidad discreta, pero otros merecerían presentarse a nuestro concurso, y a buen seguro que obtendrían un premio.

Benko nos ofrece una web que destaca especialmente por su desparpajo. Con fondos negros sobre los que directamente integra los textos y algunas notas de color, visualmente no es especialmente llamativa, aunque el *look* que posee quizá se acerque bastante a la estética de una parte de nuestra juventud.

Pablosoft se lo toma con más calma, aunque no duda en ofrecernos avances y muestras de los juegos que tiene en preparación, alguno de los cuales, como el llamado *Pablo Kopter*, se nos antoja de gran interés.

Mención aparte merece también la página de Maqnaziller, en la que se nos ofrece la posibilidad de disfrutar de su particular versión de *Diablo*. Se trata de un experto del rol, y no hay que despreciar los consejos que pueda darnos dentro de un género cada vez más popular.

Rafael María Claudín



Manía
/ especial-
o de la Red
uy atentos
gan pági-
darlas a
os demás

vegaremos
a vez apa-
vez mayor,
delante

/ sencilla.
t dedicado
página
es mandar
el que
rlo por
l sobre:

de nuestra

con más
en ofrecer-
de los jue-
ación,
mo el lla-
os antoja

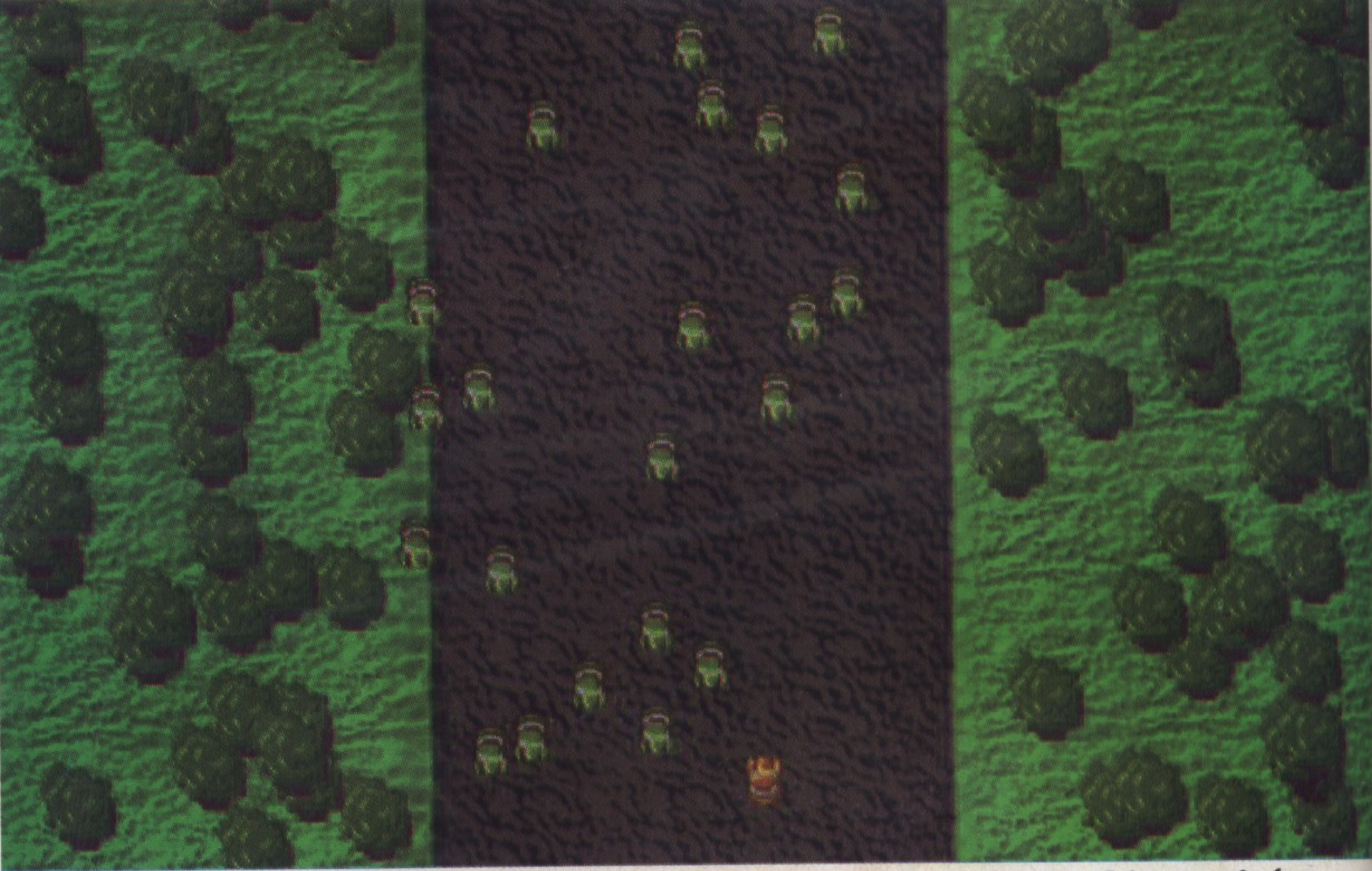
ece tam-
aziller, en
osibilidad
ular ver-
de un
/ que des-
pueda
nero cada

laría Claudín

Programando en DIV

Nuestro primer videojuego

Una vez sentadas las bases del lenguaje DIV estamos en disposición de aprender nuevos conceptos que nos ayudarán a crear juegos más vistosos y adictivos. Fondos de pantalla, control mediante joystick, punteros de ratón, colisiones y explosiones serán el menú para este número. Sin más dilaciones pasemos a programar.



Con DIV podrás realizar multitud de efectos y proco a poco el juego irá tomando forma.

En nuestro primer ejemplo veíamos como el coche se desplazaba sobre un pobre fondo negro. ¿Cómo conseguir que se mueva a través de una imagen de fondo? La solución es bien sencilla. En el fichero con los mapas del juego añadimos un nuevo mapa, ya sea cargado del disco o dibujado por nosotros, con las dimensiones de la resolución de la pantalla del juego (en nuestro caso 640x480) y le asignamos el código de mapa que deseemos (preferentemente utilizaremos los primeros números para los fondos).

Una vez realizada esta operación debemos incluir en el código de nuestro programa la siguiente línea:

```
Put_Screen(0,<código>);
```

Donde *código* es el número del código del mapa que le asignamos al fondo en el fichero. El cero se debe a una razón bien sencilla. En cualquier programa podemos cargar varios ficheros FPG a la vez. A cada uno de ellos se le asigna un número según el orden en el que son cargados empezando por el 0. Por tanto el primer parámetro de la función *put_screen()* indica el número del fichero donde debe buscar la imagen de fondo. Como en nuestro caso sólo hemos cargado un fichero, éste tiene asignado el número 0.

Es importante realizar la carga del mapa de fondo después de cargar los ficheros FPG ya que si no, no puede encontrar el fichero al que se refiere la función.

El código del anterior programa quedaría como sigue:

```
PROGRAM ejemplo1;

BEGIN
set_mode(m640x480);// línea de
comentario
load_fpg("ejemplo1.fpg");
put_screen(0,3); //el gráfico 1 y
2 estaban ya asignados a los coches
coche( );
LOOP
    FRAME;
END
END
```

Observamos la presencia de líneas de comentario en el programa. Dichas líneas no tienen otra funcionalidad que servir de apoyo a la comprensión del programa por otras personas. Su sintaxis es la siguiente:

```
// <comentario sobre el programa>
```

Todo lo escrito a partir de las dos barras es ignorado por el compilador DIV a la hora de crear el programa.

Controlando con el joystick

¿Tienes un joystick y no sabes que hacer con él? Conéctalo a tu ordenador y prueba a cambiar la función *key()* de las sentencias *if* del proceso *coche()* y sustitúyelo por las siguientes instrucciones:

```
Izquierda : joy.left
Derecha: joy.right
Arriba: joy.up
Abajo: joy.down
Botón 1: joy.button1
Botón 2: joy.button2
Botón 3: joy.button3
Botón 4: joy.button4
```

El proceso coche quedaría de la siguiente forma:

```
PROCESS coche()
BEGIN
graph=1;
x=100;
y=100;
LOOP
  IF (x>640)
     x=0;
  END
  IF (x<0)
     x=640;
  END
  IF(joy.right)
   x=x+1;
  END
```

```
    IF(joy.left)
      x=x-1;
    END
    FRAME;
  END
END
```

Ejecútalo y ya tenemos el joystick funcionando.

Puntero del ratón

En cualquier juego de estrategia, aventura gráfica y otros tantos, es necesario para su fácil control la presencia de un puntero de ratón. DIV no se queda atrás y nos ofrece la posibilidad de añadir un puntero fácilmente de la siguiente forma:

Mouse.graph = <código>

Donde *código* es el código del mapa del cursor gráfico dentro del fichero FPG.

Si lo colocamos en la parte principal del código (justo después de *put_screen()*) y ejecutamos el programa, veremos cómo aparece el puntero. Por supuesto, antes debemos cargar o dibujar un puntero y añadirlo al fichero de mapas con el código que deseemos. El tamaño del cursor puede ser cualquiera.

Tal vez alguien se pregunte, ¿Pero cómo podemos controlar el coche con el ratón? Su funcionamiento es bastante distinto al del joystick o el teclado. Ya no debemos colocar millones de sentencias *IF* para cada una de las teclas de dirección. Basta con asignar a las variables *x* e *y* del proceso que queremos controlar mediante el ratón, los valores *mouse.x* y *mouse.y*. Con esto las variables *x* e *y* seguirán la posición del ratón a lo largo de la pantalla.

Si queremos que nuestro programa realice alguna acción cuando se pulse alguno de los botones del ratón, debemos utilizar la misma estructura que utilizamos con el joystick pero con los siguientes valores:

Botón izquierdo: mouse.left
Botón derecho: mouse.right
Botón central: mouse.middle

Por tanto, si queremos que nuestro coche aparezca en la posición x=0 si pulsamos el botón izquierdo, deberemos escribir:

```
IF (mouse.left)
X=0;
END
```

Con esto finalizamos todo lo referente a los controles de los videojuegos.

Las bases de la animación. Bucles

Un muñeco que se desplace por la pantalla sin mover ni brazos ni piernas es muy poco realista. A la secuencia de imágenes que dan sensación de movimiento, actividad o dinamismo es a lo que llamamos animación. ¿No os suenan los cursores animados de Windows? No son más que una serie de imágenes que se repiten indefinidamente dando esa sensación de movimiento.

¿Cómo podemos realizar una animación en DIV? En primer lugar debemos dibujar cada una de las imágenes (*frames*) que compondrán nuestra animación. Los colocamos en el fichero de mapas en el orden en que deben ser animados. Una vez hecho esto tan solo debemos cambiar la variable *graph* del proceso que queremos animar de forma circular. Por ejemplo, si hemos asignado a los *frames* de la animación los códigos de 5 a 10, debemos darle a *graph* los valores 5,6,7,8,9,10,5,... indefinidamente.

Para realizar esto disponemos de unas estructuras denominadas *bucles*.

Un bucle es una repetición de sentencias finita o infinita. Es decir, con un bucle podemos decirle al programa que realice 5 veces una determinada acción sin tener que escribirla 5 veces seguidas. Con esto ahorramos muchísimo código y además dotamos a nuestros programas de nuevas capacidades. Las principales estructuras son:

- *LOOP...END*: repite indefinidamente las sentencias que dentro de él se encuentran. Ya fue explicado anteriormente.
- *REPEAT...UNTIL*(<condición>): repite las sentencias que en él están incluidas hasta que se cumpla la condición que está entre paréntesis. La sintaxis que hemos de seguir en la condición es la misma que la utilizada por la sentencia *IF*.
- *WHILE* (<condición>)... *DO*: igual que la anterior. La diferencia con la estructura *Repeat*, es la evaluación previa de la condición antes de ejecutar las sentencias que dentro de él se encuentran.
- *FROM* <variable>=<valor1> TO <valor2>; ... *END*: ejecuta su contenido para el valor asignado para la variable y lo repite hasta que su valor sea *valor2*, incrementando en cada repetición en 1 el valor de la variable.
- *FOR* (<inicialización>;<condición>;<incremento>)...*END*: Repite su contenido para valores de la variable inicializada desde su valor inicial hasta que deje de cumplir la condición indicada, incrementándose su valor según indica el incremento.

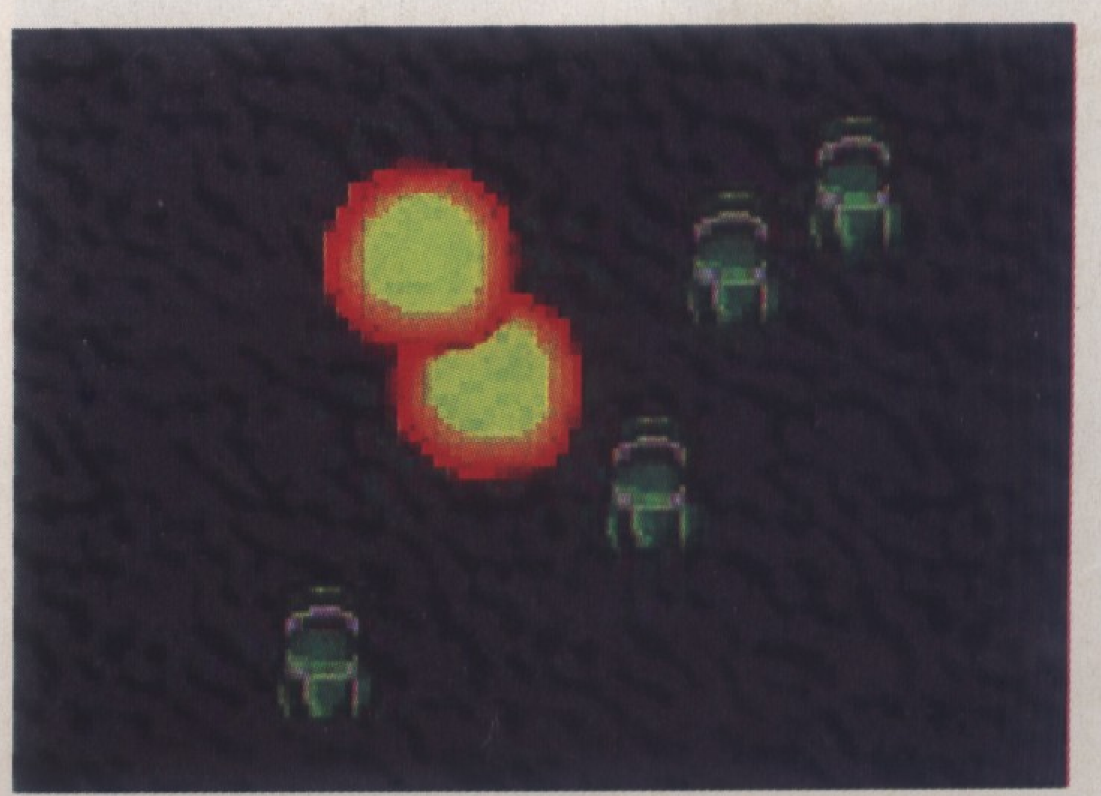
Poco a poco iremos viendo en ejemplos el uso de cada uno de los bucles.

Cabe destacar que en cualquier caso podremos utilizar indistintamente cualquier tipo de bucle, aunque siempre será recomendable el uso de un tipo determinado dependiendo de lo que pretendamos realizar.

Volviendo a las animaciones, gracias a los bucles podemos asignar a la variable *graph* dichos valores de forma circular. En primer lugar vamos a experimentar con una simple animación de explosión.

Gracias al generador de explosiones (menú de mapas) podemos fabricar explosiones de forma fácil e intuitiva. En él podremos indicar el ancho y alto de la explosión, dejaremos los que indica por defecto, así como el número de imágenes que va a tener la animación y los colores de inicio y fin. Disponemos de 3 tipos de explosiones, siendo el tipo A el más sencillo y el C el más complejo. Si pulsamos el botón aceptar, veremos cómo se abren tantas ventanas como imágenes hemos indicado generando la animación. La animación generada variará cada vez que la creemos, por lo que nunca obtendremos dos animaciones exactamente iguales. Si hacemos doble clic sobre cualquiera de las imágenes podremos editarlas. Si pulsamos la tecla *TAB* o *Mayúsculas + TAB* podremos ver la animación generada. Hay que resaltar que el generador utilizará los colores que le indiquemos, así como aquellos colores intermedios que encuentre en la paleta actual y

Un bucle es una repetición de sentencias finita o infinita que nos ahorra muchísimo código, gracias al cual podremos simplificar enormemente nuestras líneas de programación



Los efectos visuales son parte importante de todo juego que se precie.

que le faciliten la creación de la animación.

Una vez tenemos todos los *frames* de la explosión, los introducimos por orden de generación en el fichero que contiene los mapas de nuestro juego, asignándoles códigos correlativos. Ya estamos en disposición de escribir el código que mostrará la explosión de forma indefinida:

```
PROCESS coche()
BEGIN
[...]
LOOP
 FROM graph=4 TO 9;
 FRAME;
 END
END
[...]
END
```

Con esto repetimos el valor 4...9 para *graph*, con lo que asociamos en cada *frame* una imagen distinta y por tanto hemos generado la animación buscada de forma indefinida.

Destrucción de procesos

Un aspecto importante en la creación de cualquier videojuego en DIV es la destrucción de los procesos que ya no son útiles o no deben estar presentes en el programa.

Un proceso deja de existir cuando se llega a la sentencia *END* de su código. Hasta ahora nuestro programa ejecutaba indefinidamente una serie de sentencias que se encontraban dentro del bucle *LOOP...END*, pero nunca llegaba al final de la ejecución. Para conseguir que se salga del bucle lo que debemos hacer es incluir una sentencia *BREAK*, que indica al compilador cuándo debe salirse de la ejecución del bucle para continuar con el resto de sentencias.

DIV posee un generador de explosiones muy sencillo y cómodo para el programador, de manera que podremos resolver fácilmente un aspecto muy importante del juego

Con el generador de explosiones de DIV, podemos crear fácilmente este tipo de efectos.

Por lo tanto, si queremos hacer que algún proceso desaparezca una vez que se haya salido de la pantalla, entonces tan sólo debemos comprobar que sus coordenadas no sobrepasen los límites de ésta. Esto lo haríamos de la siguiente forma:

```
PROCESS coche()
BEGIN
Graph = 1;
X=320;
Y=240;
LOOP
 IF (x>640)
 BREAK;
 END
 FRAME;
END
// aquí podemos realizar una animación de explosión para simular la destrucción.
 END
```

Dentro del programa que tenemos entre manos conseguiremos hacer que se destruya el coche si sobrepasa el límite derecho de la pantalla. Como se indica en el programa, podemos hacer que antes de alcanzar el final de la ejecución del proceso se genere una animación de explosión. Para ello utilizaríamos la sentencia *FROM...TO* de esta forma:

```
[...]
FROM graph = 4 TO 9;
FRAME;
END
```

Con esto logramos la explosión deseada. Una forma de conseguirla muy sencilla pero, al mismo tiempo, muy efectiva.

Realización de las colisiones de un juego

Un aspecto casi obligatorio en todos los juegos, sobre todo los arcades y juegos de acción pero también en muchos otros, es la detección de colisiones entre distintos procesos que se estén realizando. ¿Cómo conseguimos que nuestro programa nos indique cuándo dos coches se chocan? La forma es mucho más sencilla de lo que podríamos pensar en un primer momento. Así, tenemos dos formas distintas de hacerlo. Una mediante tipos de procesos y otra mediante identificadores. En este artículo trataremos sólo el primer caso, por ser el más sencillo y más que suficiente para que nuestro propósito.

Este tipo de detección consiste en localizar cuándo nuestro proceso colisiona con otro proceso de un determinado tipo. Para ello, utilizaremos la función *collision()*. Su funcionamiento es sencillo. Devuelve un valor nulo si el proceso que estamos analizando no ha colisionado con el tipo indicado y el código de identificador del proceso con el que se colisiona en este último caso. Por tanto le aplicaremos una sentencia *IF* a dicha función para detectar las posibles colisiones y su consecuencia de la forma que sigue:

```
IF (collision(TYPE <proceso>))
<sentencias>
END
```

Podemos observar la presencia de una nueva palabra reservada: *TYPE*. Esta palabra reservada nos devuelve una variable asociada a un tipo de proceso que indicamos con el nombre de dicho proceso en el código. Por tanto, si queremos que dicho proceso sea del tipo coche, tan sólo deberemos escribir:

```
IF (collision (TYPE coche)) ...END
```

Si le añadimos una sentencia *BREAK* en su interior, y estando incluido dentro del bucle *LOOP...END* de un proceso cualquiera, podemos hacer que dicho proceso se destruya. Aún así hay un método mucho más elegante y legible que consiste en realizarlo a través de un bucle *REPEAT...UNTIL*:

```
PROCESS coche()
BEGIN
Graph=2;
X=320;
Y=240;
 REPEAT
 // aquí moveríamos el coche
 FRAME;
 UNTIL(collision (TYPE coche))
 FROM graph=4 TO 9;
 FRAME;
 END
 END
```

Con esto detectaríamos la colisión con otro coche, pero para ello hemos de crear un nuevo proceso del tipo coche. Dicha creación es tan sencilla como añadir una línea *coche()* en el código principal del programa. De esta manera podremos crear en cualquier momento un nuevo proceso llamando a la función desde cualquier punto del programa y bajo cualquier circunstancia.

Por fin nuestro primer juego

Tras tantas páginas de duro aprendizaje, por fin disponemos de un amplio abanico de posibilidades

con el que jugaremos para poder crear nuestro primer arcade más o menos serio, y sobre todo jugable. Para ello vamos a emplear todo lo que hemos aprendido a lo largo de estos dos artículos.

En este primer juego manejaremos un coche a lo largo de una carretera e intentaremos no chocarnos con los diferentes vehículos que vienen de frente. Para ello, situaremos un fondo de pantalla donde tendremos una carretera con dos laterales de bosque. Para simular un desplazamiento del fondo de la pantalla, crearemos una serie de árboles que irán apareciendo desde lo alto de la pantalla para desaparecer cuando lleguen a la parte inferior, pero sólo se desplazará por la zona de bosque, donde nuestro coche no podrá entrar.

Limitaremos, de este modo, el área de movimiento del coche a la zona de la carretera e impediremos que sobrepase el mismo centro de la pantalla. Los coches se moverán hacia abajo y hacia nuestro coche, perdiéndose la partida cuando nuestro coche colisione con otro.

Hemos creado tres distintos tipos de proceso, a saber: *planta*, *coche* y *coche2*. Estudiaremos con más detalle el funcionamiento de cada uno de ellos:

- *Proceso coche*: será el proceso que obedecerá nuestras instrucciones con el control mediante los mismos cursores. Cuando colisiona con un proceso del tipo *coche2*, entonces se destruye y mostramos una animación de explosión. Su movimiento está limitado a un área determinada y, como podemos observar, el desplazamiento hacia atrás es inferior al desplazamiento hacia delante. Con esto se pretende que nunca retroceda a la misma velocidad a la que avanzan los árboles, de forma que dé una mayor sensación de movimiento.
- *Proceso planta*: genera una planta en los lados de la carretera, en una posición aleatoria. Para ello tendremos que recurrir a la función *rand(inico,fin)* que devuelve un valor aleatorio entre los valores inicio y fin. Podemos ver la utilización de la sentencia *REPEAT...UNTIL* para poder hacer desaparecer la planta una vez que haya atravesado el borde inferior de la pantalla.
- *Proceso coche2*: Su funcionamiento es idéntico al que hemos podido constatar con el proceso planta, sólo que aparece en la sección de pantalla de la carretera y se destruye cuando choca con el proceso coche o con uno de su mismo tipo. Para poder detectar la colisión entre dos o más tipos de procesos, recurrimos a la palabra reservada *OR* (o), que devuelve *cierto* en una condición cuando una de las dos proposiciones o las dos proposiciones son ciertas. Este tipo de expresiones lógicas las estudiaremos en profundidad más adelante.

La creación de los procesos *planta* y *coche2* se realiza dentro del bucle principal, donde hemos colocado una condición aleatoria para la aparición de éstos. Este método es bastante sencillo, pero puede complicarse aún más si queremos limitar el número de procesos o simplemente queremos realizar una creación algo más racional.

Un aspecto que es bastante importante dentro del programa es la presencia de la sentencia *let_me_alone()*. Esta función destruye todos los procesos que están funcionando actualmente en nuestro juego, de forma que el bucle principal termina su ejecución al no encontrar más procesos en ejecución, dando el programa por finalizado.

```
PROGRAM ejemplo3;

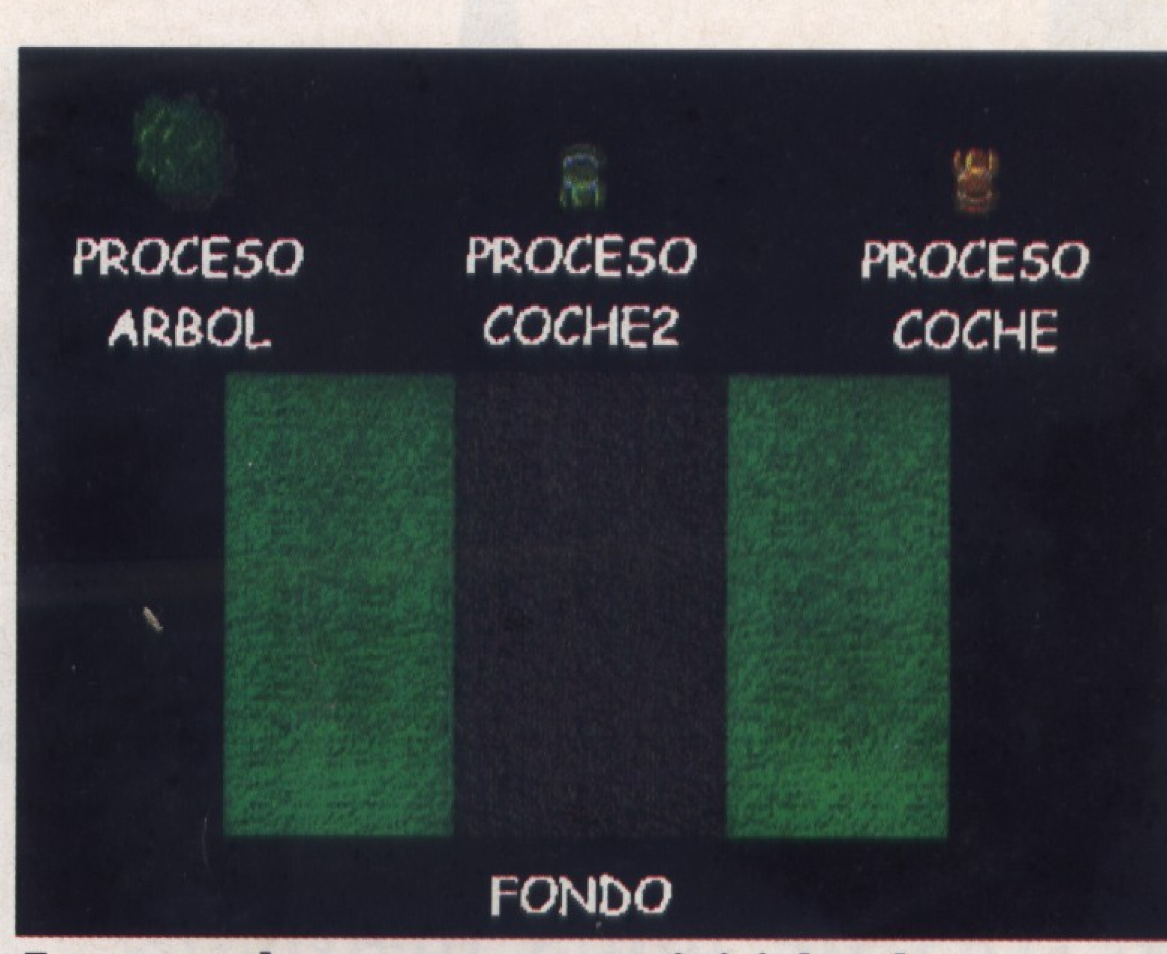
BEGIN
set_mode(m640x480);
load_fpg("c:\div\divmania\ejemplo3.fpg");
put_screen(0,1);

coche();
LOOP
   IF (rand(0,5)>=4)
planta();END
   IF (rand(0,8)>=7)
coche2();END
   FRAME;
END
END

PROCESS coche()
BEGIN
graph=2;
x=320;
y=400;
 REPEAT
   IF (x>440)
      x=440;
   END
   IF (x<200)
      x=200;
   END
   IF (y>465)
      y=465;
   END
   IF (y<350)
      y=350;
   END
   IF(key(_right))
    x=x+2;
   END
   IF(key(_left))
     x=x-2;
   END
   IF(key(_up))
     y=y-2;
   END
   IF(key(_down))
     y=y+1;
   END
   FRAME;
 UNTIL (collision(TYPE coche2))
 FROM graph=5 TO 10; FRAME;
END
 let_me_alone();
 END

 PROCESS planta()
 BEGIN
 graph=3;
 x=rand(0,180)+(440+22)*rand(
0,1);
  y=-22;
 REPEAT
   y+=2;
   FRAME;
 UNTIL (y>490)
 END

 PROCESS coche2()
 BEGIN
 graph=4;
 x=rand(190,430);
 y=0;
   REPEAT
    y+=3;
    FRAME;
   UNTIL(collision(TYPE coche)
OR collision (TYPE coche2))
   FROM graph=5 TO 10;
FRAME; END
 END
```



Estos son los tres procesos iniciales de nuestro programa.

Pablo Trinidad

Funciones no visuales

Al otro lado del monitor

Detrás de unos gráficos impactantes y un juego adictivo, siempre hay una gran cantidad de procesos y funciones no visuales, que es lo que le da a un juego ese toque de elegancia que le desmarca de entre los demás. En un primer paso analizaremos el módulo ASCII, que hará las delicias sobre todo de los programadores de aventuras gráficas.



DIV Games Studio se está consolidando como una de los más importantes entornos de desarrollo de videojuegos.

Este mes daremos un paso más en la creación de *dlls* de funciones analizando el módulo ASCII.dll, de Antonio Marchal, que cubre el vacío que dejó DIV en la manipulación de archivos de texto. Además completaremos la librería de funciones matemáticas que iniciamos el mes anterior. Comenzamos:

Matemáticas en DIV

Aunque a simple vista parecen ser poco útiles, pueden resolver más de un problema al programador, sobre todo en el mundo de la demoscene. Por supuesto si queremos realizar una rotación de un punto en la pantalla (aunque sea un tanto engorroso realizarlo en DIV) debemos recurrir a las funciones trigonométricas. Por ello, y en previsión de la posible necesidad de algún programador desesperado, completaremos este módulo de funciones matemáticas que incluye cálculo de senos, cosenos, tangentes y logaritmos neperianos.

ASCII contiene funciones que permiten cargar,grabar y manipular ficheros de texto en formato ASCII así como crear variables de texto

El mes pasado, y para facilitar la comprensión, la función de cálculo de senos debía recibir el ángulo en radianes. ¿Qué debemos hacer para que realice las operaciones con grados? Simplemente debemos dividir el ángulo entre 2PI y multiplicarlo por 360. Pero para ello necesitamos el valor de PI, que lo obtendremos declarándolo como una constante mediante *#define*. El código de la función quedará por tanto implementado de la siguiente forma:

```
#include "stdio.h"
#include "math.h"

#define GLOBALS
#define PI 3.1416

#include "div.h"

void seno()
{
int angulo=getparm();

int
resultado=sin(angulo*2*PI/(1000*36
0))*1000;

retval(resultado);
}
```

El resto de funciones han sido desarrolladas de forma análoga. Para más información, debemos acudir al archivo de explicación incluido con el archivo *math.zip*, que podremos encontrarlo en el CD de la revista o en la dirección *http://members.xoom.com/pablot*

ASCII, útil ante todo

Una vez finalizado el desarrollo de nuestra *dll* de funciones matemáticas, vamos a pasar a analizar una de las librerías de funciones que más ha llamado la atención en el mundo de DIV. ASCII contiene funciones que nos permite cargar, grabar y manipular ficheros de texto en formato ASCII (que se engloba dentro del primer módulo de funciones llamado ASCII), así como crear variables de texto donde se podrán introducir datos desde el teclado (formando parte del módulo INPUT).

Para que podáis ir disfrutando de las funciones que nos proporciona este módulo, aquí tenemos una breve descripción de las distintas opciones de que dispone:

- *ASCII_VERSION():* retorna la versión del módulo de funciones. La última versión es la 2.
- *ASCII_LOAD(<fichero>,<offset variable>,<linea1>,<linea2>):*

carga en la variable de texto que indiquemos (debemos enviar la dirección de ésta mediante el operador &) el contenido del fichero indicado (nombre del archivo) entre las líneas 1 y 2.

- *ASCII_SAVE(<fichero>,<texto>, <modo>):* graba al fichero indicado el texto enviado a la función. El modo valdrá 0 si queremos añadir dicho texto al final del fichero y 1 si queremos que nos cree un nuevo fichero.
- *ASCII_ADD(<texto1>, <texto2>):* crea y devuelve una nueva variable que contiene a ambos textos unidos.
- *ASCII_COPY(<fichero>,<inicio>,<longitud>):* crea y devuelve una nueva variable que contiene el texto del fichero indicado la posición de inicio desde donde se va a comenzar la lectura y la cantidad de caracteres a leer (longitud).
- *ASCII_PASTE(<texto1>, <texto2>,<posición>):* pega el texto 2 sobre el texto 1 a partir de la posición indicada ignorando el contenido sobreescrito.
- *ASCII_DUP(<valor>,<cantidad>):* devuelve y crea una variable de texto del tamaño indicado por cantidad y que contiene el carácter indicado por valor tantas veces como cantidad.
- *ASCII_FREE(<identificador>):* libera la memoria asignada a las variables de texto creadas por las funciones *add*, *copy* y *dup*. Es recomendable su uso cuando ya no se utilicen más estas variables.
- *ASCII_LEN(<texto>):* devuelve el tamaño en caracteres del texto cuyo identificador hemos enviado a la función.
- *ASCII_GETCHAR(<texto>,<posición>):* devuelve el carácter leído en la posición indicada del archivo de texto.
- *ASCII_SETCHAR(<texto>,<posición>,<carácter>):* escribe el carácter indicado en la posición del fichero de texto.
- *INPUT_VERSION():* devuelve la versión actual del módulo *Input*. La última versión es la 1.
- *NPUT_NEW(<tamaño>):* devuelve el identificador de una nueva variable de *Input* cuyo tamaño será el indicado para la función.
- *INPUT_ADD(<identificador>, <carácter>):* añade al final de la variable cuyo identificador enviamos el valor indicado por carácter. No deben incluirse en las variables códigos de control como pueden ser *Enter* o borrado. Por ello debe comprobarse el valor a introducir antes de añadirlo.
- *INPUT_DELETE(<identificador>):* borra el último carácter introducido en la variable de *Input* indicada por identificador.
- *INPUT_FREE(<identificador>):* libera la memoria reservada para una variable del tipo *Input*.

Algunas funciones de DIV para las Dlls

El código de las *dlls* de DIV está compilado en 32 bits con modo protegido (acceso lineal a la memoria, permitiendo un direccionamiento máximo de 4 Gb) mediante el módulo DPMI.

Algunas funciones, como el manejo de ficheros y sobre todo la asignación dinámica de memoria que tienen un comportamiento distinto en modo real que en el protegido, y la utilización de funciones estándar como *malloc(), free(), fopen(), fclose()*, pueden provocar fallos graves en el entorno. Por ello, hemos de utilizar estas funciones especiales que vienen a sustituir a las estándar de ANSI C:

Asignación dinámica de memoria:

- *EXTERN void *(*div_malloc) (size_t __size)*: utilizar en lugar de malloc().
- *EXTERN void (*div_free)(void *__ptr):* utilizar en lugar de free()

Apertura y cierre de ficheros:

- *EXTERN FILE *(*div_fopen)(char *,char *):* utilizar en lugar de fopen().
- *EXTERN void (*div_close)(FILE *):* utilizar en lugar de fclose().

Otras funciones:

- *EXTERN int (*div_rand)(int rang_low,int rang_hi):* devuelve un valor aleatorio dentro del rango.
- *EXTERN void (*div_text_out) (char *texto, int x, int y):* imprime el texto en la posición de la pantalla indicada.

El resto de funciones de manipulación de ficheros pueden utilizarse sin obtener errores.

Acceso a los datos de un programa

Todos los datos que se manipulan y contienen en un programa se almacenan en la memoria. La forma de acceder a ellos en DIV es bien sencilla y sin duda efectiva. Partimos de una dirección de memoria base. En función a ésta, cualquier proceso, datos, textos,... tienen una posición relativa. Por tanto, si queremos acceder a la dirección de memoria donde se almacena un determinado dato, basta con conocer la dirección base y el desplazamiento de dicho dato.

A la hora de programar *dlls*, esto se traduce a la manipulación de un *array* llamado ***mem[]***. La posición *mem[0]* es la posición base de la memoria, y a través de distintos valores podemos acceder a los datos de texto, procesos y otros.

Cuando reservamos memoria dinámicamente, obtenemos como puntero el desplazamiento respecto a la memoria base.

El código de las Dlls de DIV está compilado en 32 bits con modo protegido, es decir, el acceso lineal a la memoria, permitiendo un direccionamiento máximo de 4 Gb, y mediante el módulo DPMI

Los principales desplazamientos de que disponemos son los siguientes:

- *stack*: índice de la pila.
- *text_offset:* índice donde comienzan los textos.
- *id_offset:* índice del último proceso ejecutado.
- *id_init_offset:* índice del primer proceso.
- *id_start_offset:* índice del proceso principal
- *id_end_offset:* índice del último proceso.

Pues bien, en la librería ASCII, pretendemos manipular datos de texto situados en memoria y para ello utilizaremos el desplazamiento *text_offset*. Podremos obtener la dirección de los datos de texto enviando en primer lugar la dirección (mediante el operador «&») y tomando su valor con *getparm()*. La dirección así obtenida, no es una dirección relativa a la posición base, sino respecto al comienzo del *buffer* de datos. Por tanto para acceder a un determinado dato lo haremos mediante la siguiente sentencia:

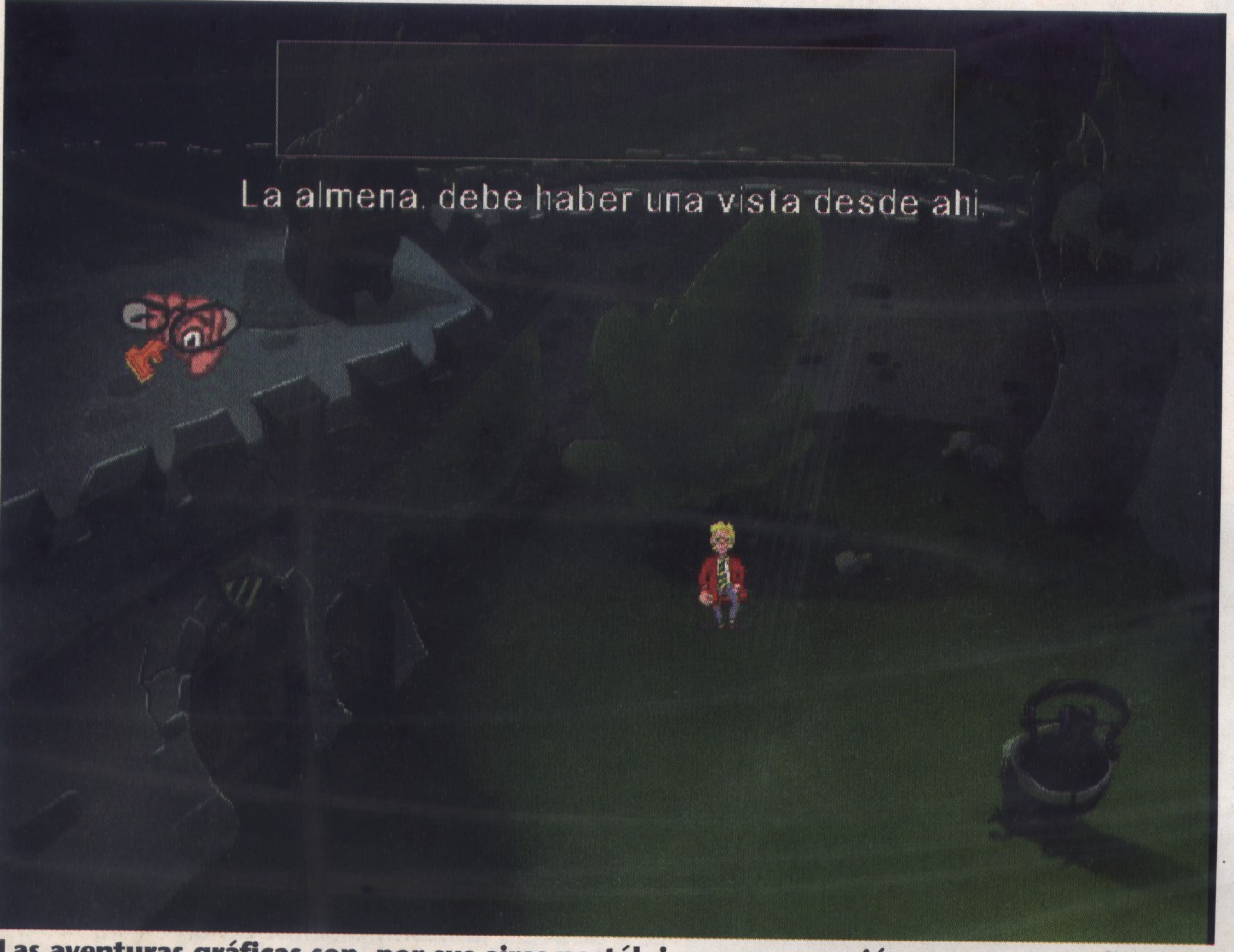
mem[text_offset+dato];

Funciones de math.dll

- SENO(ángulo): devuelve el seno del ángulo expresado en grados.
- COSENO(ángulo): devuelve el coseno del ángulo expresado en grados.
- TANGENTE(ángulo); devuelve la tangente del ángulo expresado en grados y –1 si el ángulo es 90° y por tanto su tangente infinita.
- LN(valor): devuelve el logaritmo neperiano del valor expresado. Si el valor es 0 o un número negativo, la función devuelve –1.

El resultado devuelto por las distintas operaciones están expresados en coma fija con 3 decimales, es decir 45668 es igual a 45,668.





Las aventuras gráficas son, por sus aires nostálgicos y su conexión con otros medios, uno de los géneros más populares.

En el anterior número hablamos de *getparm()* como una función que recogía los datos enviados a la función, pero no reseñamos que el orden de recepción de estos datos es inverso, es decir, el último dato enviado es el primero en recibirse, debido a su estructura de pila. Además, hemos de tener en cuenta que *getparm()* siempre devuelve un *int* de 32 bits, por lo que deberemos realizar una conversión de tipos siempre que el parámetro pedido sea de otro tipo.

El acceso a los datos de un programa desde DIV es bien sencillo y sin duda efectivo; en función de una dirección de memoria base cualquier proceso, datos, textos,etc. tiene una posición relativa

Analizando el código

Para ir asimilando conceptos, vamos a ver el funcionamiento de la función *ASCII_SETCHAR()* realizando un análisis exhaustivo. El código de la función es el siguiente:

```
void ascii_setchar()
{
 char *texto1;

 int variable2=getparm();
 int posiciondeltexto=getparm();
 int variable1=getparm();
 texto1=(char *)&mem[variable1+text_offset];

 if (posiciondeltexto<0) retval(-1);
 if (posiciondeltexto>strlen
 (texto1)) retval(-1);
 texto1[posiciondeltexto]=(char)variable2;
 retval(1);
}
```

El código se divide en 3 partes diferenciadas:

En una primera parte, declaramos las variables de los parámetros de entrada y además un puntero de carácter, que es el que va a contener la cadena en la que vamos a escribir el carácter contenido en *variable2*. Podemos observar cómo el valor contenido en *texto1*, es la dirección de memoria de la *variable1*, que hemos obtenido, como comentábamos anteriormente, mediante desplazamientos relativos. Al asignar dicha dirección, cualquier cambio realizado en la variable *texto1*, afectará directamente al contenido de la memoria, es decir, al texto.

Posteriormente comprobamos que la posición donde queremos colocar el nuevo carácter sea positiva y no supere el tamaño de la cadena. En caso de que se incumpla, devuelve un valor negativo de error.

Por último, hechas las comprobaciones, estamos ya en disposición de escribir la *variable2* en la posición de memoria indicada y devolver el valor 1, que indica que la operación ha sido realizada con éxito, no sin antes haber realizado la conversión a *char* (ya que el valor obtenido por *getparm()* era un *int*).

Como podemos observar, el código necesita de unos conocimientos claros de punteros y conversión de tipos.

Esta función puede servir como modelo para la comprensión de las funciones *paste*, *getchar* y *len*. En el resto de funciones de ASCII, se añade un nuevo concepto que es la asignación dinámica de memoria. Para comprenderlo, vamos a analizar la función *ASCII_DUP()* por ser la más sencilla:

```
int *nuevopunteroascii,mitxt; //
variables globales

void ascii_dup(){
    int primertxt;
    int nuevoascii;
    int *empiezatexto;
    int *realpunteroascii;
    char *texto1;
    int cantidaddetexto=get-
   parm();
    int caracterelegido=get-
   parm();

    // A reservar memoria....
    nuevopunteroascii=(int
    *)div_malloc(cantidadde-
    texto+8);
    mitxt=(int)nuevopunteroas-
    cii;

    //y calculamos el offset del
   nuevo puntero

 empiezatexto=&mem[text_off-
 set];
    primertxt=(int)empiezatex-
    to;
    mitxt+=3;
    mitxt&=-4;

    // puntero colocado....
    // Guarda valores inter-
    nos....
    realpunteroascii=(int *)
    mitxt;

 *realpunteroascii=MARCA_CADE-
 NA_TIZO_ASCII ;
    realpunteroascii++;
    mitxt=(int)realpunteroascii;

    nuevoascii=(mitxt-pri-
    mertxt)/4;
    char * ptr2;
    ptr2=(char*)mitxt;
    int contador;
    for (contador=0;conta-
    dor<cantidaddetexto;conta-
    dor++){
ptr2[contador]=(char)caractere-
legido;
    }
    ptr2[cantidaddetexto]=0;
    retval(nuevoascii);
}
```

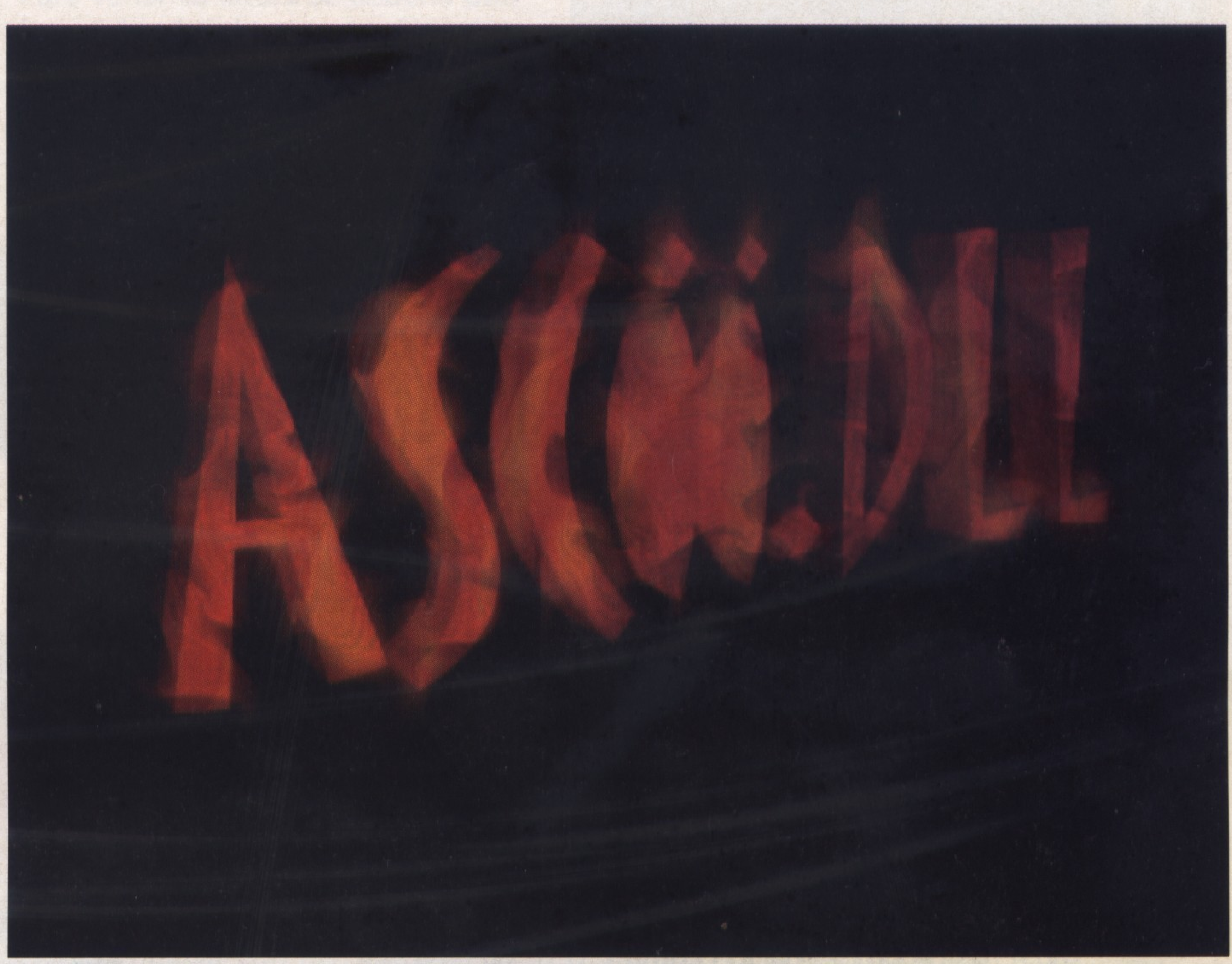
En esta función se crea una nueva variable de texto, reserván-

dose la memoria necesaria para el tamaño indicado (cada carácter ocupará 1 byte) y además reservamos 8 bytes más, 4 para la marca distintiva de las variables de texto creadas por ASCII y cuyo valor es *MARCA_CADENA_TIZO_ASCII*, 3 para el redondeo de la dirección a un múltiplo de 4 y otro que corresponde al carácter nulo necesario para indicar el final de la cadena. Pero para que DIV no considere la marca como un carácter de la propia cadena, se devuelve como puntero el segundo carácter, es decir, el primero de la cadena, por lo que DIV desconocerá la existencia de dicha marca y por tanto no afectará al resto de operaciones. Entonces, ¿Para qué utiliza la marca? Si queremos liberar la memoria asignada anteriormente a una variable creada por la propia librería, debemos saber que ésta fue creada en ella misma y no por DIV, ya que sino podría provocar un error grave en el programa. Por tanto, antes de liberar memoria, se comprueba que la cadena haya sido creada verificando dicha marca.

Pasemos a analizarla de una forma un poco más detallada. Las variables definidas y su correspondiente funcionalidad son:

- primertxt: entero con la dirección de *empiezatexto
- mitxt: entero con la dirección de *nuevopunteroascii
- nuevoascii: entero con la dirección de la nueva variable de texto. Es el que se enviará a DIV como dirección de dicha variable.
- *nuevopunteroascii: contiene la dirección de memoria de la memoria asignada para la variable nueva.
- *empiezatexto: contiene la dirección del comienzo de text_offset
- *realpunteroascii: puntero de tipo entero utilizado para escribir la marca de ASCII.
- *ptr2: puntero de tipo carácter utilizado para escribir en la cadena el carácter indicado tantas veces como contenga la variable cantidaddetexto.
- *texto1:No utilizada en la función.

Una vez hayamos realizado este análisis, estaremos en disposición de entender completamente el proceso. En primer lugar reservamos la cantidad de memoria que necesitaremos, y que se corresponde al tamaño de la cadena más 8 (la razón de que esto sea así la explicamos con detalle anteriormente).



ASCII sigue siendo una de las más importantes librerías de funciones, muy útiles para realizar trabajos con DIV; es importante conocer las distintas opciones que nos ofrece.

Posteriormente tomamos la dirección base de las variables de texto que tenemos. Esta dirección la almacenamos en el entero *mitxt*, que manipularemos de la manera indicada en el propio código. Dicha manipulación no es más que un redondeo del entero a 4, para más tarde poder dividirlo por dicho número. El porqué de esta necesidad viene de la manipulación de direcciones de memoria en DIV. DIV almacena todas las direcciones como enteros divididos entre 4 (desplazamiento hacia la derecha de 2 bits), por lo que cualquier dirección nueva que obtengamos debe ser devuelta en dicho formato.

Por eso mismo, la dirección contenida en *nuevoascii* la obtenemos calculando la posición relativa de la nueva memoria reservada respecto al *buffer* de textos y dividiéndola con cuatro, por lo que DIV ya no tendrá problemas a la hora de acceder a todos los datos que creemos.

Por último, tan sólo tendremos que escribir los datos dentro de la memoria, pero aparece entonces un nuevo problema. En primer lugar, deseamos escribir la marca, que es un entero de 4 bytes, por lo que utilizaremos un puntero de tipo *int *realpunteroascii* y después un puntero de tipo *char *ptr2*, que utilizaremos para escribir ya la cadena final.

Debemos observar que para poder finalizar la cadena tiene que aparecer un carácter nulo, que se le asigna mediante una orden muy sencilla, a saber:

ptr2[cantidaddetexto]=0;

Y también para que no aparezca la marca como carácter, debemos desplazar la dirección asignada para el texto 4 bytes, con lo que ya está solucionado el problema.

Conclusión

Pues con esto ya tenemos toda la base teórica suficiente para comprender el funcionamiento de esta magnífica librería así como dotarle de nuevas capacidades o mejorar las funciones. El resto de funciones son afines y no entrañarán dificultad a la hora de intentar comprenderlas.

El código se divide en tres partes: declaras las variables de los parámetros, compruebas la posición en la que se colocará el nuevo caracter y escribes la *variable2* en la posición de memoria indicada

Por supuesto, quisiera agradecer a Antonio Marchal (Tizo) la cesión de este archivo para el uso de todos los usuarios, así como su gran dedicación al mundo de la programación en DIV. Gracias.

Y por si a alguien le quedaba alguna duda, podéis escribir un E-mail a la dirección *trinidad@arrakis.es* o visitar la dirección *http://members.xoom.com/pablot*. Cualquier sugerencia o duda acerca de esta sección será bien recibida.

Hasta el próximo número.

Pablo Trinidad

3D/Red

La estrella de los arcades

¿Hay algo más importante que los gráficos en un videojuego? En este artículo aprenderemos todo lo que hay que tener en cuenta para crear este aspecto.

En este número el lector, poco o nada familiarizado con lenguajes de programación como C o C++, va a poder aprender los conceptos básicos sobre cómo crear un juego en 3D en una plataforma un tanto limitada en este aspecto como es el Div Games Studio.

¿Qué tipo de juego voy a hacer?

Dependiendo del tipo de juego la estructura del programa puede variar, dado que puede ser de carreras, plataformas, lucha, acción... o un arcade en primera persona como el archiconocido Doom que, aunque es casi imposible recrearlo al 100% sin el uso de librerías externas (DLL's), se puede llegar a crear un tipo de juego muy parecido, con paredes no en 3D sino mediante *sprites*. De hecho, ya se ha creado un juego así con Div. En este número el tipo de juego será un arcade/plataformas/acción.

A la hora de crear los gráficos de nuestro juego un factor muy importante son los tipos de sprites que utilizamos

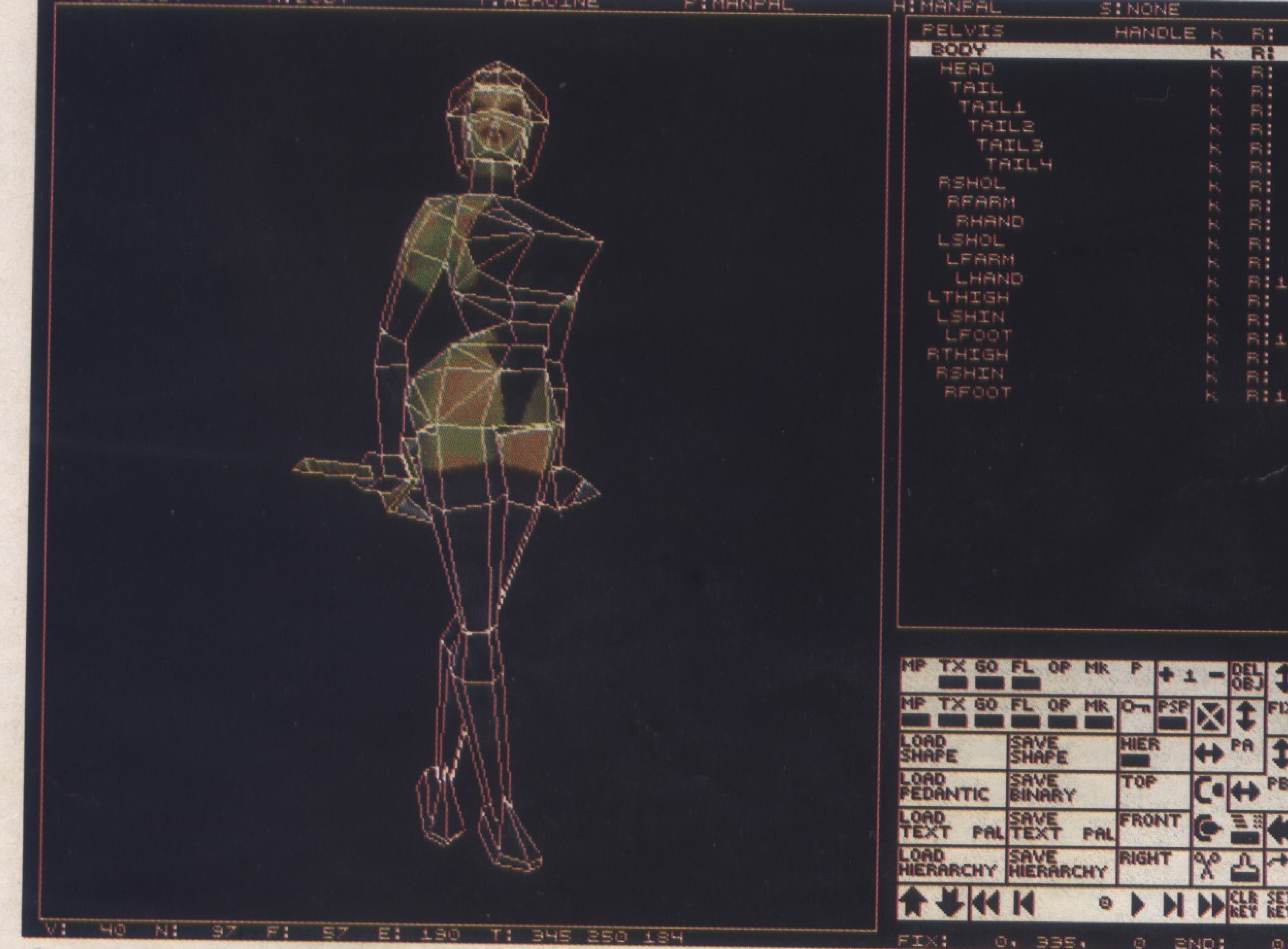
Preparando los gráficos

A la hora de crear los gráficos, un factor muy importante a tener en cuenta es que se deben hacer de tipo *sprite*. Es decir, en vez de hacer texturas para aplicarlas a polígonos como se haría normalmente en cualquier juego 3D, hay que crear los dibujos enteros de cómo sería el objeto en cuestión. Si se tiene en cuenta que hay que dibujarlos desde distintos puntos de vista para que giren, la solución menos engorrosa para crear los gráficos es usar un programa como el *Autodesk 3D Studio*, ya que, tras diseñar el modelo, se pueden obtener todas las vistas de forma casi automática. También se pueden crear *sprites* fijos, que no giren. Es decir, que se miren desde el ángulo que se miren, siempre permanecen iguales. Estos *sprites* son usados, por ejemplo, en el Doom para objetos fijos.

Entrando en la programación

A la hora de empezar a programar, es necesario usar el *modo7*, ya que es la única herramienta para trabajar en 3D que tiene Div. La estructura básica del *modo7* es la siguiente:

- *m7.camera*: esta variable indica qué proceso está siguiendo la cámara del *modo7* que se indica. Por ejemplo, para asignar la cámara al proceso actual, se tendría que poner *m7.camera=id*; y para asignarla a un proceso cuyo identificador es personaje, sería *m7.camera=personaje.* Esta variable podemos modificarla para sí, por ejemplo, podemos hacer que la cámara siga a un misil o a un enemigo.
- *m7.height*: indica la altura a la que se encuentra la cámara. Cuanto más alta esté, más definido se verá el suelo, pero también más pequeño. Esta variable debería ser siempre la misma que la *height* local del proceso al que sigue la cámara para que no desaparezca de pantalla por encima o por debajo.
- *m7.distance*: es la distancia de la cámara al píxel exacto del *modo7* sobre el que se encuentra el proceso al que sigue la cámara. Si es un valor bajo, al variar el ángulo del proceso que sigue la cámara, ésta girará sobre sí misma, mien-



Con Autodesk 3D Studio se pueden obtener todos las vistas automáticamente.

tras que al aumentarlo girará alrededor de ese proceso, pero siempre apuntando hacia él.

- *m7.horizon*: es la altura de la línea del horizonte del mapa en *modo7*. Ésta se puede usar para mirar abajo y arriba, aumentándola o disminuyéndola. Hay que tener en cuenta que si hubiera un proceso de un objeto "volando", que permanece a una altura mayor a la del personaje, y se mira arriba, el objeto seguiría en la misma posición. Por eso, a la vez que se varía el horizonte, hay que hacer que los objetos bajen una determinada cantidad su *height* (altura).
- *m7.focus*: controla el focal de la cámara. Es una variable muy útil para conseguir efectos interesantes de luces. Para ello se ajusta a un valor muy bajo (de 2 a 5) y se hace girar el ángulo de la cámara, siempre que el mapa del *modo7* tenga unos colores apropiados.

En realidad, la función de esta variable es lograr efectos de perspectiva; por ejemplo acercando el personaje y reduciendo el focus de la cámara se logra un clásico efecto cinematográfico.

- *m7.z*: es el plano de profundidad del *modo7*, normalmente 256. Se usa como la *z* normal de los procesos.
- *m7.color*: es el color que tiene el fondo del *modo7* en el que se acaba el gráfico, es decir, el color de relleno de aquellos sitios del *modo7* con coordenadas fuera del gráfico.

La única textura que se utilizará es la del suelo, que se aplicará al *modo7*. Para una mejor calidad y para evitar pixelaciones, se recomienda crear un gráfico grande y aumentar la altura de la cámara.

Para obtener un combate de un juego de lucha con cámara trasera, por ejemplo, se deberían seguir los siguientes pasos.

- Crear una región de *modo7* en pantalla. El horizonte debería tener un valor sobre 85, que cae más o menos por el centro de la pantalla.
- Llamar a los procesos que ponen el marcador de vida, victorias, energía, etcétera.
- Llamar a los procesos de los personajes.

Si fuera para un solo jugador, la cámara seguiría al personaje controlado por el jugador.

A la hora de especificar coordenadas en los procesos, hay que fijarse que éstos no sean fijos de pantalla sino del escenario 3D. Su variable *local C_TPYE* debe ser *C_M7* para que tome la *X* y la *Y* del *modo7*.

Para variar su altura se debería usar la variable *height*, que es sólo para *modo7*.

Otro factor a tener en cuenta es que el mejor método de movimientos en el *modo7* es el de *angle* y *advance*. Aunque, se puede hacer mediante aumentos de *X* y de *Y*, es demasiado costoso y con resultados poco ventajosos.

Cómo conseguir sacar los ángulos de los personajes

Cuando el enemigo mira hacia un lado y el jugador está detrás, el enemigo tiene que aparecer de espaldas, y no de lado como indica su ángulo. Para evitar esto hay que realizar unas pequeñas comprobaciones en el bucle principal del proceso del personaje.

Lo que se debe hacer es coger el ángulo del enemigo hacia el proceso del personaje y, según sea el ángulo actual y el obtenido, asignarle un gráfico. Pero lo malo es que al trabajar con *get_angle* suele dar valores demasiado altos. Para eso se puede usar este bucle *repeat* al principio del bucle del personaje principal:

```
IF (angle_obt<=0 OR
angle_obt>360000)
REPEAT
IF (angle_obt<=0);
angle+=360000;END;
IF (angle_obt>360000); angle-
=360000;END;
UNTIL (angle_obt>0 AND
angle_obt<=360000)
```

Lo que hace este bucle es mantener la variable *angle_obt* entre 1 y 360000 para que sea más fácil trabajar con ella. También es muy útil hacerlo con todas las variables de los ángulos.

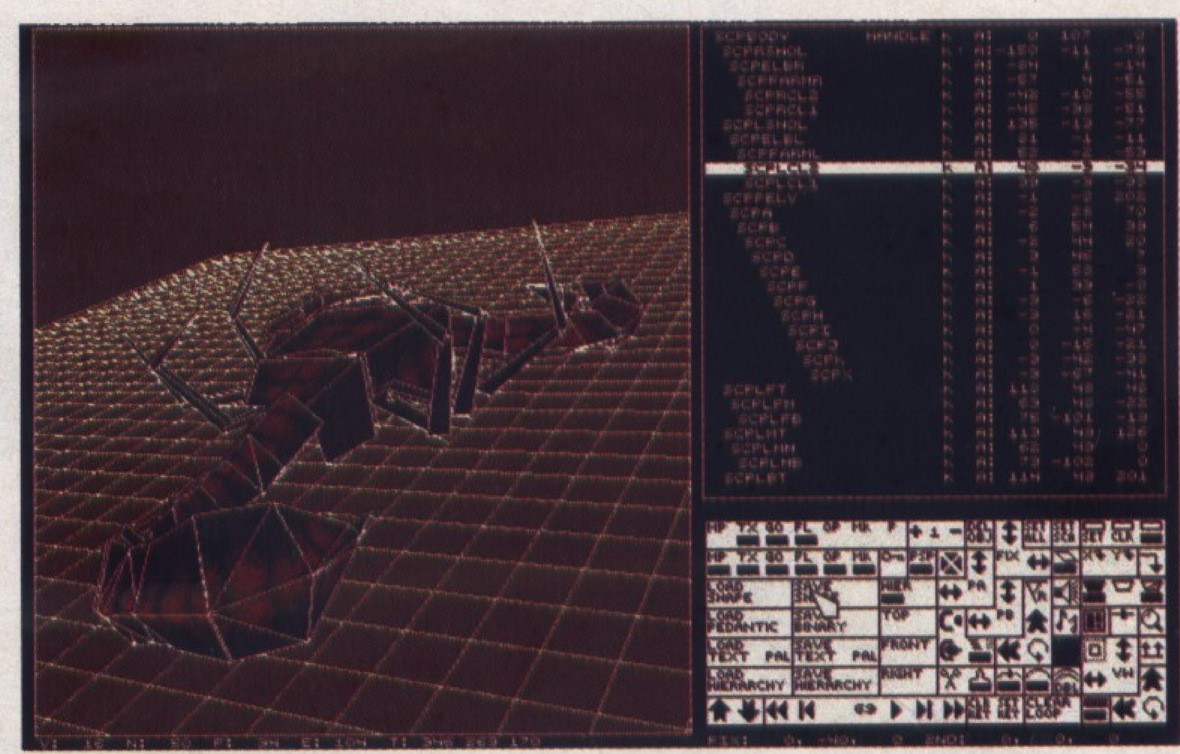
Otros puntos importantes

Esto es más o menos lo básico para hacer un juego en 3D con Div, aunque luego siempre se pueden añadir elementos como paredes (que siempre serán por *sprites*, a menos que se use una *DLL*), objetos, etc.

La ambientación es un detalle que corre a cuenta del autor del programa, pero a continuación se citan algunos efectos que consiguen mejorar mucho el programa, ocultando la limitación de Div en el aspecto 3D. Estos efectos los ha usado el autor de este artículo en un juego de lucha 3D con Div:

- Cambios en el mapa del *modo7*: cuando ocurran explosiones, caídas, colisiones fuertes contra el suelo, un efecto muy bueno es usar un *map_xput* con un gráfico de rotura/grietas sobre el gráfico del *modo7*, para "estropear" la zona sobre la que se ha colisionado.
- Terremotos de pantalla: uno de los efectos más vistosos es hacer vibrar la pantalla cuando tengan lugar golpes fuertes. Se consigue moviendo al azar las *X, Y y height* de los procesos, incluyendo la *height* del *modo7*, el horizonte... unas cuantas veces, y luego restableciéndolas.
- *Fades* de pantalla: aprovechar el *fade* de colores de Div es otro método para conseguir un efecto interesante: hacer un *fade* rápido a blanco y luego restaurarlo podría acompañar a un disparo, por ejemplo.
- Giros de cámara: es complicado de conseguir, pero se obtiene más calidad al hacer que la cámara gire alrededor del personaje según sea necesario. Lo más fácil con Div sería que lo hiciera sólo en ocasiones especiales (ataque especial, muerte...)

Ferminho



Región de modo 7 en pantalla.



Se obtiene más calidad al hacer que la cámara gire alrededor del personaje.



La ambientación del juego corre por cuenta del autor.



La interfaz gráfica

Preparando la interactividad

Siguiendo nuestro guión prepararemos el corazón de nuestra aventura y sentaremos las bases para su programación.

En el artículo anterior definimos cómo va a ser nuestra primera aventura gráfica. Ahora debemos decidir la forma en que el jugador podrá indicar qué cosas quiere realizar en cada momento.
Los buenos jugadores de aventuras saben lo muy variados que pueden ser las interfaces de usuario de una aventura gráfica. Esta interfaz es la que permite al jugador comunicarse con él. Muchos jugadores sienten debilidad por el formato clásico, «al antiguo», como coloquialmente se le llama. Este formato consiste en dividir la pantalla en al menos dos partes. La parte principal (más de dos tercios de la pantalla) es el juego en sí. En esta zona aparecen los escenarios y es en ella donde se desarrolla la acción. La segunda suele encontrarse dividida en dos más. Una para las acciones posibles (mirar, ir a, abrir...) y otra para representar la bolsa de objetos, donde aparecerá todo los objetos que portamos.

> El cursor es una de las partes más esenciales de la interfaz y habrá que prestarle una atención especial

Siguiendo nuestro guión del juego, notamos que las acciones a realizar por el jugador serán:

- *Ir a/hacia* lugar. Para que nuestro protagonista pueda recorrer los escenarios del juego
- *Hablar* con personaje. Esta opción nos permitirá dialogar con otros personajes de la aventura como nuestro compañero, el jardinero o los niños.
- *Mover* objeto. Necesitaremos mover un cuadro del pasillo para recuperar la llave de la habitación.
- *Mirar* objeto. Nos permitirá obtener más información acerca de las cosas mirándolas con atención.
- *Coger* objeto. A menudo necesitaremos que nuestro personaje coja objetos y los lleve de un sitio a otro para poder usarlos convenientemente.
- *Dar* objeto *a* personaje. Con esta acción podremos darle a otro personaje objetos que lleve nuestro protagonista, como por ejemplo, la pelota a los niños.
- *Abrir* objeto. Generalmente nos servirá para abrir puertas.
- *Cerrar* objeto. Como abrir, nos servirá para las puertas.
- *Usar* objeto. Esta acción nos permite usar un único objeto apropiadamente.
- *Usar* objeto *con* objeto. Nos permite combinar el uso de objetos o usar un objeto sobre otro.

Diagrama del formato clásico

Zona de escenarios

Línea de opciones

Menú de Opciones   Bolas de Objetos

Definiendo la interfaz

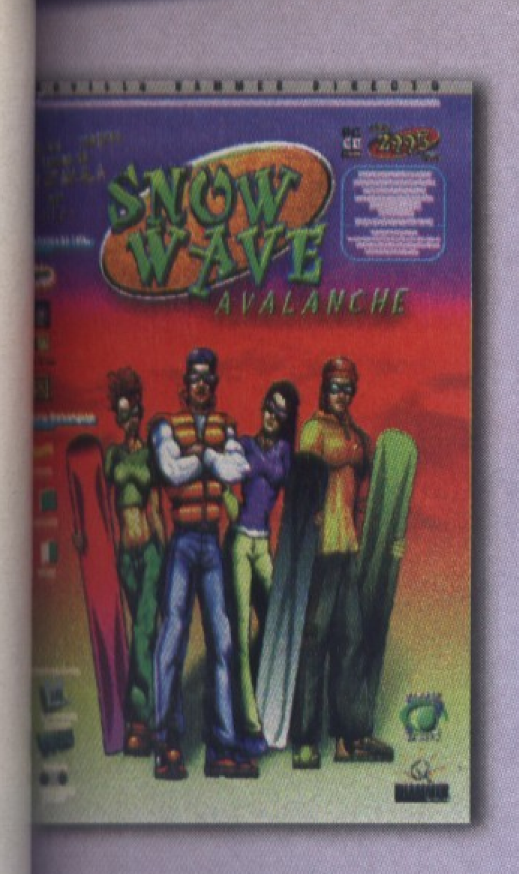
Nuestra interfaz va a ser esencialmente clásica. Más de los dos tercios de pantalla se destinan a escenarios, el resto a control de acciones. El control de acciones se divide en tres partes: línea de acción (arriba), menú de acciones (abajo izquierda) y bolsa de objetos (abajo derecha).
En el *cuadro 1* podemos apreciar la distribución final de la pantalla. La zona de escenarios y la bolsa de objetos los trataremos más adelante. De momento sólo hablaremos del menú de opciones y la línea de acción. La línea de opciones nos va a indicar en todo momento lo que estamos a punto de hacer. Las acciones a realizar por nuestro personaje las seleccionaremos en el menú de opciones, de momento. Para hacérselo más fácil al jugador, existirán una serie de acciones por defecto que no necesita seleccionar para realizar. Estas acciones dependerán del objeto en sí, por ejemplo, la acción por defecto para una puerta cerrada será abrirla.

Código para activar el puntero del ratón y controlarlo desde el teclado

```
// Asignamos el gráfico al puntero del ratón
mouse.file = f_general;
// Fichero gráfico
mouse.graph=1; // Activa cursor
// **** Código del programa ****
// **** Bucle principal ****
IF (key(_right)) mouse.x+=10;
END;
IF (key(_left)) mouse.x-=10;
END;
IF (key(_up))    mouse.y-=10;
END;
IF (key(_down)) mouse.y+=10;
END;
IF (key(_space))
mouse.left=TRUE; END;
IF (key(_enter))
mouse.right=TRUE; END;
// **** Resto del bucle principal
****
// **** Resto del código del
programa ****
```

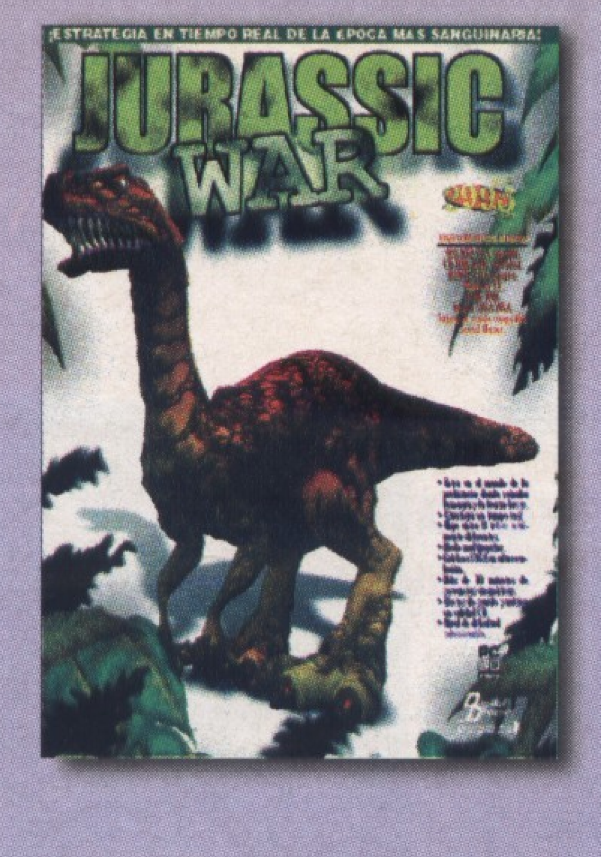
El cursor

Aunque hemos hablado bastante de la interfaz gráfica, no hemos hablado aún de lo más importante de ella: el cursor. El cursor es quién va a permitir realizar todo en nuestra aventura. Hoy, nos bastaría con hacer que el cursor se moviese con el ratón, usando el puntero del mismo. Es aconsejable permitir controlarlo también desde el teclado, no es difícil de hacer y siempre hay quién nos lo agradecerá. Esto en DIV es muy sencillo de conseguir.
Asignamos al puntero del ratón un gráfico que previamente hemos realizado. En este momento nuestro gráfico ya se desplaza por toda la



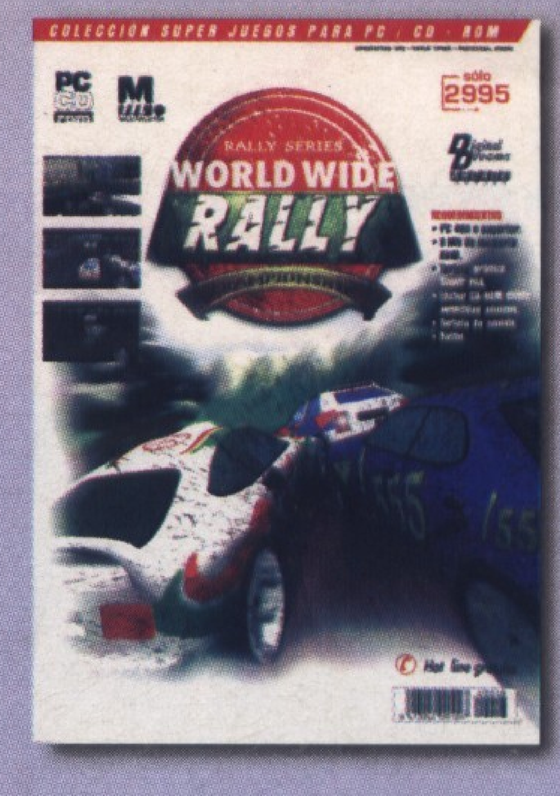
Disfruta con Snow Wave Avalanche de vertiginosos descensos entre riscos y cañones helados, realizando aterradores saltos al vacío e imposibles acrobacias dignas de un auténtico campeón.

2.995 ptas.
Incluye CD-ROM.



Jurassic War es el primer juego de estrategia en tiempo real que te conduce a la prehistoria. Elige tu tribu y lucha por el dominio de la isla en una época donde la emoción y los peligros acechan en cualquier momento.

2.995 ptas.
Incluye CD-ROM.



Un arcade de carreras en 3D que hará las delicias de todos los *fitipaldis*. Gracias a su alta resolución, a su soporte multijugador y a sus 15 cuidadas pistas podrás sentirte un auténtico piloto al volante de tu coche preferido.

2.995 ptas.
Incluye CD-ROM.



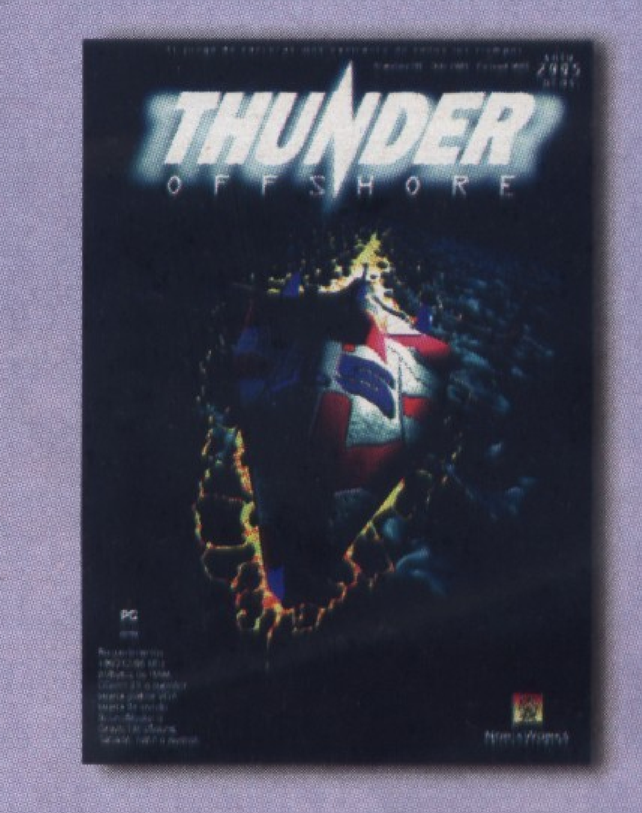
La segunda parte de este completo paquete de los juegos más espectaculares y adictivos de los últimos tiempos. En un solo paquete, toda la adicción de un juego de estrategia, la diversión de un pinball y la acción de un arcade de carreras con lanchas motoras.

2.995 ptas.
Incluye CD-ROM.



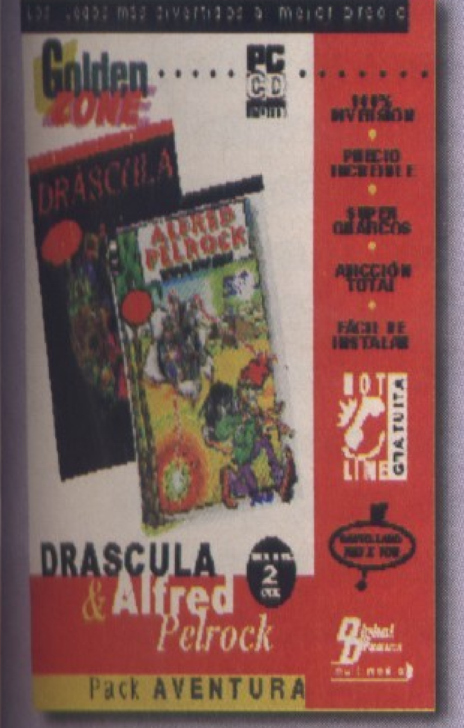
El juego de tenis más espectacular. El simulador 3D definitivo de tenis. Con increíbles efectos visuales como destellos de sol, sombras dinámicas, fuentes de luz y cámaras móviles que seguirán la acción en todo momento.

2.995 ptas.
Incluye CD-ROM.



Modelos 3D voxelizados de geometrías superiores a 50 mil polígonos, paisajes fotorrealistas con resolución automática y superficies reflectantes gracias a la técnica VTP. 3 Equipos, 6 thunder Arrows y 15 circuitos diferentes.

2.995 ptas.
Incluye CD-ROM.



Incluye Dráscula y Alfred Pelrock, dos aventuras gráficas. La primera desarrollada en Transilvania y la segunda en el Antiguo Egipto.

1.995 ptas.
Incluye CD-ROM.

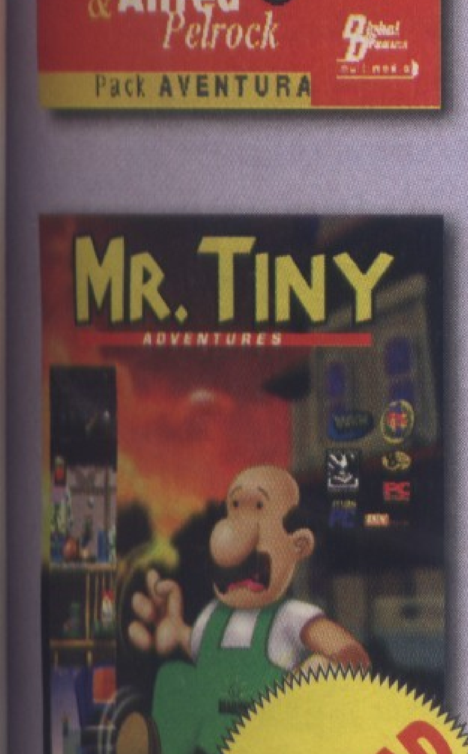


Seis estupendos programas al precio de uno. Incluye juegos de estretegia, fútbol, rally, rol, arcade y tenis.

1.995 ptas.
Incluye CD-ROM.

Permite crear juegos comerciales y libres de royalties. Posee un entorno integrado que incluye un diseñador gráfico, generador de fuentes de letras y explosiones, así como 15 juegos y multitud de tutoriales.

4.995 ptas.
Incluye CD-ROM.

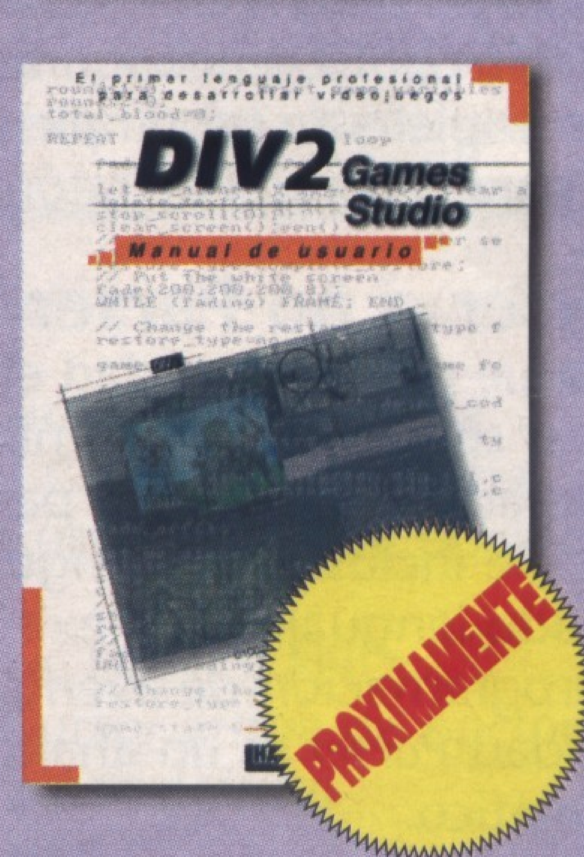


El nuevo héroe de las plataformas ha llegado a tu PC. No pierdas la oportunidad de guiar a Mr. Tiny por las más divertidas aventuras que jamás hayas visto.

2.995 ptas.
Incluye CD-ROM.

Tokenkai nos sumerge en una sociedad futurista gobernada por los intereses económicos donde la pobreza y el crimen imperan en las calles. Tan sólo eliminando al líder, Rojas, conseguirás tu objetivo: acabar con el cártel de narcotráfico más grande del mundo.

2.995 ptas.
Incluye CD-ROM.



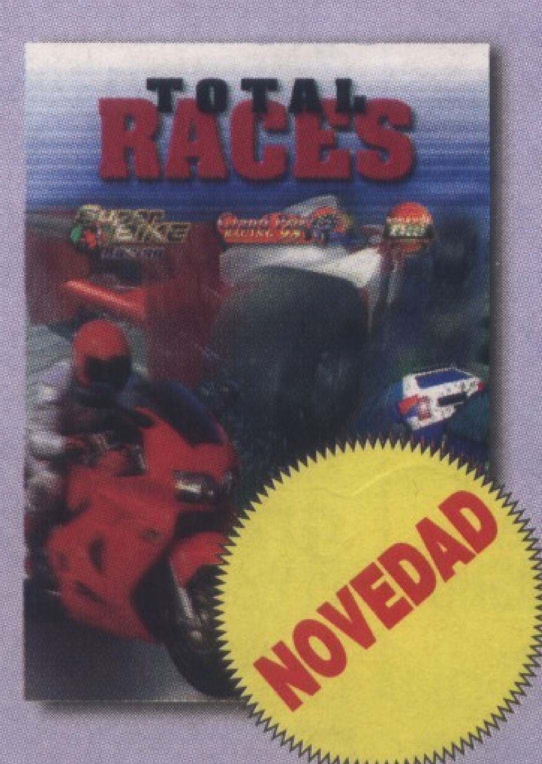
Herramienta para crear juegos comerciales. Posee editor de texto mejorado. Mundo 3D. Juegos por Red. Código optimizado para Pentium. Cuenta con 15 juegos y tutoriales para un fácil manejo.

4.995 ptas.
Incluye CD-ROM.



Contiene ocho de los mejores juegos de todos los géneros existentes: Arcade, Aventura, Deportivo, Simulador... Una colección imprescindible repartida en 2 CD-Roms.

2.995 ptas.
Incluye 2 CD-ROM.



Te ofrece tres juegos con los que la velocidad pasará a formar parte de tu vocabulario habitual. Super Bike Racing, Gran Prix Racing 98 y World Wide Rally.

2.995 ptas.
Incluye CD-ROM.



Solicite su ejemplar enviando este cupón por correo, por Fax: 91 304.17.97 o llamando al teléfono 91 304.06.22 de 9:00 a 19:00h.

❑ Sí, deseo realizar el siguiente pedido:

JUEGOS

❑ SNOW WAVE AVALANCHE	2.995	❑ TIE BREAK	2.995	❑ DIV GAMES STUDIO	4.995
❑ JURASIC WAR	2.995	❑ THUNDER OFF SHORE	2.995	❑ TOKENKAI	2.995
❑ WORLD WIDE RALLY	2.995	❑ PACK AVENTURAS	1.995	❑ DIV2	4.995
❑ TOTAL GAMES2	2.995	❑ PACK DEPORTES	1.995	❑ TOTAL GAMES	2.995
		❑ DIV GAMES STUDIO	4.995	❑ TOTAL RACES	2.995

Nombre y apellidos .... Domicilio .... Población ....
Provincia .... CP .... E-mail .... DNI/NIF ....

FORMA DE PAGO
❑ Talón a DIGITAL DREAMS MULTIMEDIA ❑ Contra-reembolso
❑ Giro postal nº .... de fecha ....
❑ Tarjeta de crédito ❑ VISA nº ❑ AMERICAN EXPRESS nº ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐
❑ Fecha de caducidad de la tarjeta .... Nombre del titular, si es distinto ....

Firma,

Rellena este cupón y envíalo a: DIGITAL DREAMS MULTIMEDIA C/ Alfonso Gómez,42 Nave 1-1-2 28037 Madrid.



La saga Monkey Island se ha convertido en todo un clásico de los juegos de aventuras gráficas.

La estructura *Accion* es el corazón de nuestra aventura gráfica

STRUCT Accion;

id_t_accion;	Usado para *write* y *delete_text*
t_accion;	Texto de la línea de acción a escribir
id_accion;	Código de la acción actual
seleccionada;	Verdadero si la acción está seleccionada
id_obj_primario;	Identificador del objeto primario
id_obj_secundario;	Identificador del objeto secundario
nivel_obj;	Indica el último objeto seleccionado
END;	

pantalla controlado por el ratón. Unas pocas líneas de código nos permitirán que el cursor también sea controlable desde el teclado con las teclas cursoras; la barra espaciadora hará la función del botón izquierdo del ratón y *RETURN* actuará como el botón derecho. En nuestra primera aventura usaremos como imagen del puntero una simple cruceta. Dependiendo del argumento y la ambientación se podría usar un gráfico más acorde, por ejemplo, una varita mágica si nuestro personaje es un mago. En el archivo *fuente_0.prg* del CD encontrarás el código completo de un simple programa para el control de ratón.

La interfaz es la parte más delicada de la programación y uno de los elementos más importantes del juego

Parser o no Parser

Aunque en la actualidad el término *Parser* se usa para denominar a los generadores de código para aventuras, e incluso a los lenguajes y pseudo lenguajes que existen para la programación de aventuras, en realidad *Parser* es un analizador gramático.

En las aventuras conversacionales, donde el texto era el rey, la interpretación de las instrucciones las realizaba el *Parser*. Dependiendo del *parser*, el jugador podía insertar instrucciones más o menos complejas, desde un simple "acariciar gato" hasta un "acariciar suavemente al lindo gatito que ronronea". El uso de un *parser* complejo y de un amplio vocabulario permitía, mediante el uso de palabras sinónimas y expresiones equivalentes, realizar la misma acción de diferentes formas: ir hacia la puerta, ir a puerta, acercarse a la puerta, caminar hasta la puerta y tantas otras combinaciones tan sólo limitadas por la memoria del ordenador y, sobre todo, la paciencia del programador. En una aventura gráfica, donde el texto pierde toda su importancia, el jugador cuenta con unas acciones muy limitadas y bien definidas, que además no se teclean. Hablar entonces de un *parser* parece no tener sentido.

Hemos definido perfectamente las acciones a realizar: *Ir, Hablar, Mover, Mirar, Coger, Dar, Abrir, Cerrar* y *Usar.* De la misma manera que hemos definido que podremos *Usar teléfono* o también *Usar monedas con máquina de refrescos.* Esto significa que a medida que el jugador interactúa con la interfaz gráfica, deberemos "construir" *la línea de acciones* que corresponda. Esto es exactamente lo contrario a lo que realiza un *parser*. De ahora en adelante, al código encargado de esta tarea lo llamaremos "Constructor gramático". Por motivos históricos, aunque el término no sea del todo correcto, llamaremos "Parser" al código que nos permite identificar, catalogar y controlar las acciones que el jugador vaya indicando.

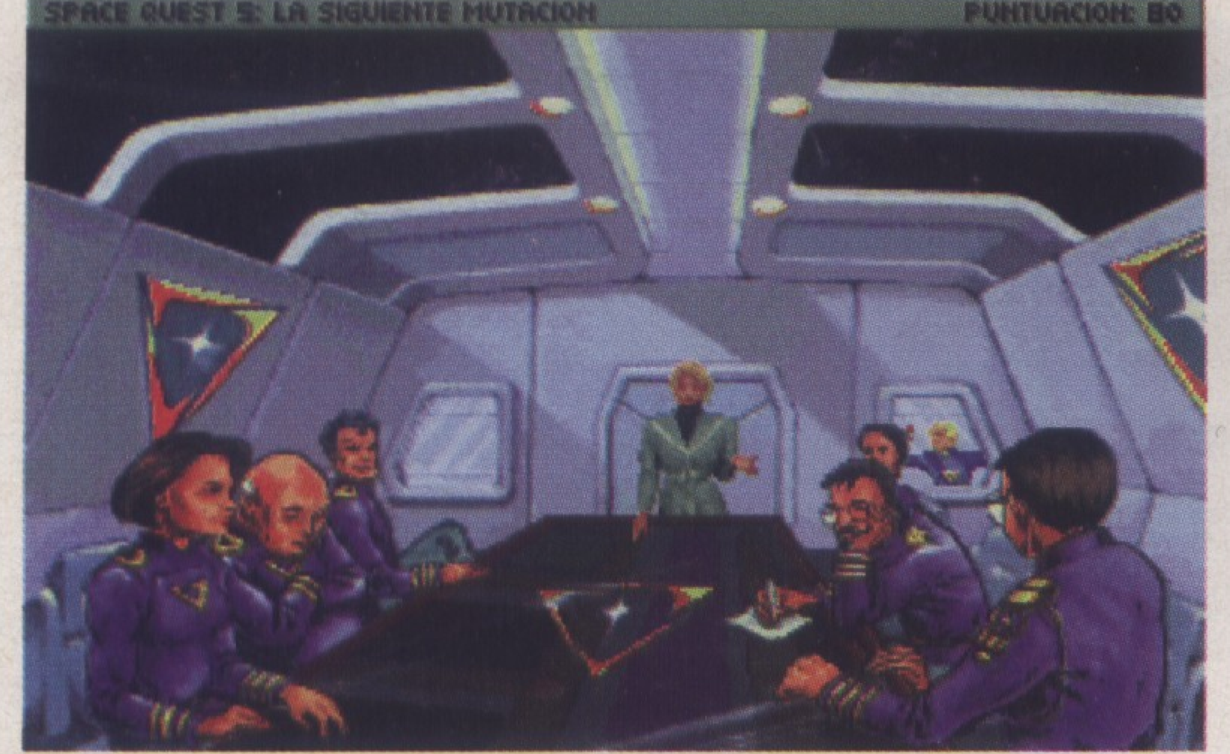
El parser y el constructor gramatical

El corazón de nuestra aventura gráfica será nuestro *parser* y nuestro *constructor gramatical.* Consistirán en un conjunto de procesos que girarán entorno a la estructura *Accion.* Esta es, quizás, la parte más delicada de la programación. De un buen *parser* dependerá el éxito y facilidad de programación de nuestro juego. Diseñaremos uno bastante sencillo, pero que cubra nuestras necesidades al completo. En primer lugar nuestro *parser* debe saber diferenciar una *acción,* un *objeto primario* sobre el que realizar la acción y, en algunos casos, un *objeto secundario.*

También tendremos que distinguir si la acción está seleccionada o no (*Accion.seleccionada*), y asegurarnos que existe un *objeto primario* antes de intentar usar un *objeto secundario* (*Accion.nivel_obj*). Los procesos que forman nuestro *parser* son:

- *sel_accion*: Selecciona temporalmente o fija la selección de una acción..
- *sel_objeto*: Selecciona temporalmente o fija la selección de un objeto.
- *des_objeto*: Anula la selección de objeto.
- *haz_accion*: Ejecuta la acción a realizar.

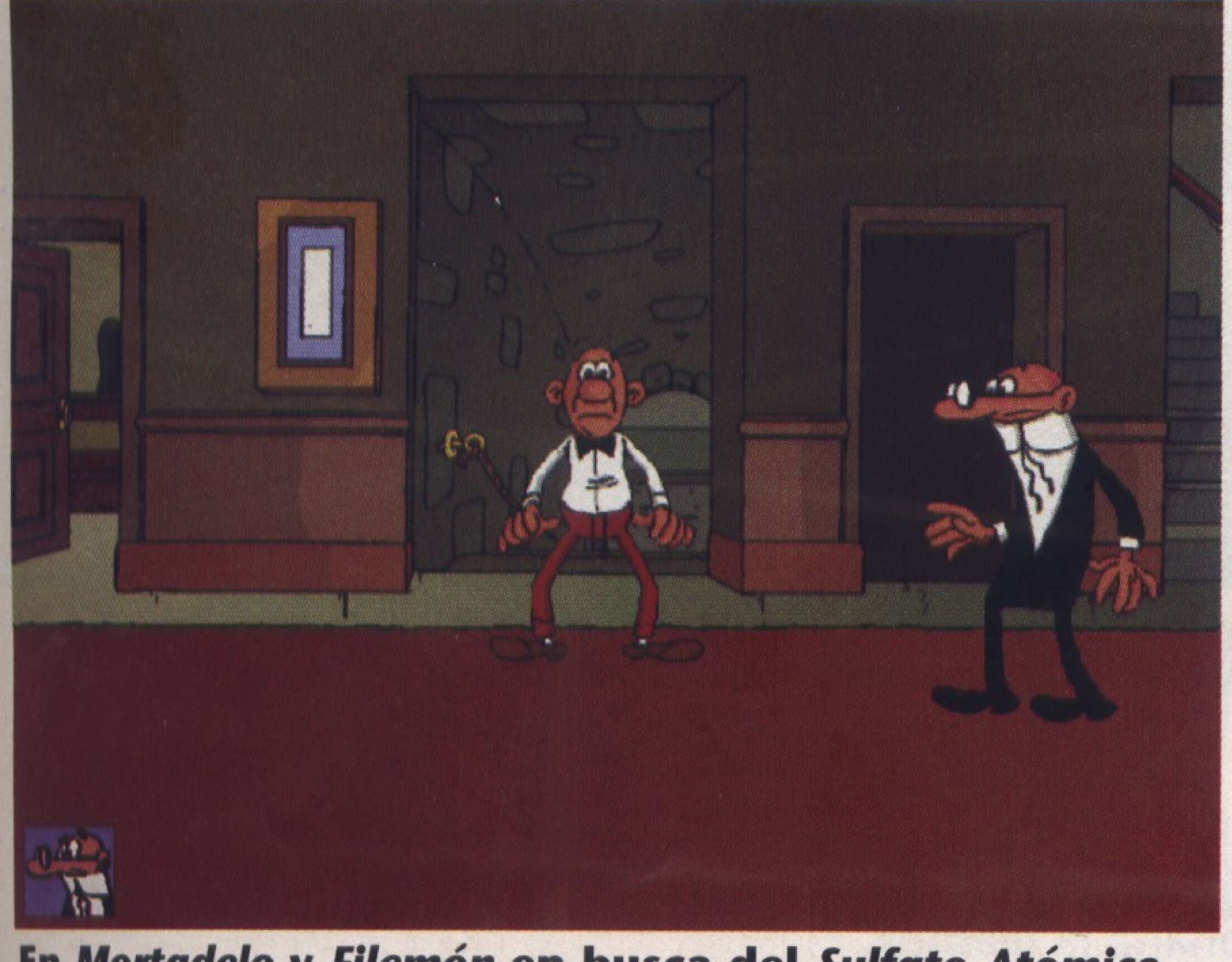
El constructor gramatical está directamente ligado al *parser*. Aprovechando la estructura *Accion,* la usaremos para conservar el texto de la línea de acción (*Accion.t_accion*) y su identificación (*Accion.id_t_accion*) para poder borrarlo una vez escrito. La tarea de



Los juegos de Sierra siempre han destacado por sus menús de iconos ocultos.

Estructura para control de menú de opciones

STRUCT Columna[2]	
x;	Inicio de la columna
xx;	Fin de la columna
STRUCT Fila[2]	
y;	Inicio de la fila
yy;	Fin de la fila
id_t;	Texto de la opción
t;	Opción
Opcion;	
END;	
END;	



En *Mortadelo* y *Filemón* en busca del *Sulfato Atómico*, además del menú de opciones y de objetos, disponemos de una opción especial para cambiar de personaje. Cada vez son más populares los juegos con múltiples protagonistas.

Código de la función *sel_accion*

```
PROCESS sel_accion(ident, select);
BEGIN
Accion.seleccionada = select;
Accion.id_accion = ident;
SWITCH (Accion.id_accion)
CASE o_none: Accion.t_accion = ""; END;
CASE o_Ir_a: Accion.t_accion = "Ir"; END;
CASE o_Hablar: Accion.t_accion = "Hablar"; END;
CASE o_Mover: Accion.t_accion = "Mover"; END;
CASE o_Mirar: Accion.t_accion = "Mirar"; END;
CASE o_Coger: Accion.t_accion = "Coger"; END;
CASE o_Dar: Accion.t_accion = "Dar"; END;
CASE o_Abrir: Accion.t_accion = "Abrir"; END;
CASE o_Cerrar: Accion.t_accion = "Cerrar"; END;
CASE o_Usar: Accion.t_accion = "Usar"; END;
END;
//    haz_accion();
END
```

realizar la construcción gramatical la desempeñará el proceso *pon_accion*.

El menú

Por fin hemos llegado al gran momento de teclear. Aunque hacer el menú parece algo trivial, definir el *parser* y el constructor gramatical antes de implementarlo ha sido un movimiento crucial para asentar las sólidas bases en desarrollo de nuestro juego.

Nuestras nueve acciones serán distribuidas en tres columnas de tres filas. Definiremos una rejilla imaginaria de nueve celdas, a las que a cada una corresponderá una opción.

Crearemos una estructura global *Columna[]* en la que especificaremos:

- *Columna[]*: Ancho de cada columna.
- *Fila[]*: Alto de cada fila, texto de la fila (opción), *Id* del texto y la constante *opcion* que corresponda.

El menú será dibujado con la fuente de letra escogida por el proceso *pon_menu*.

El constructor gramatical

El constructor gramatical lo estudiaremos con más detenimiento cuando veamos los objetos. Por el momento, nuestra función *pon_accion* no merece gran explicación debido a su sencillez. Simplemente debemos saber que será un proceso que continuamente dibujará, por ahora, el texto de la acción (*Accion.t_accion*)

El parser

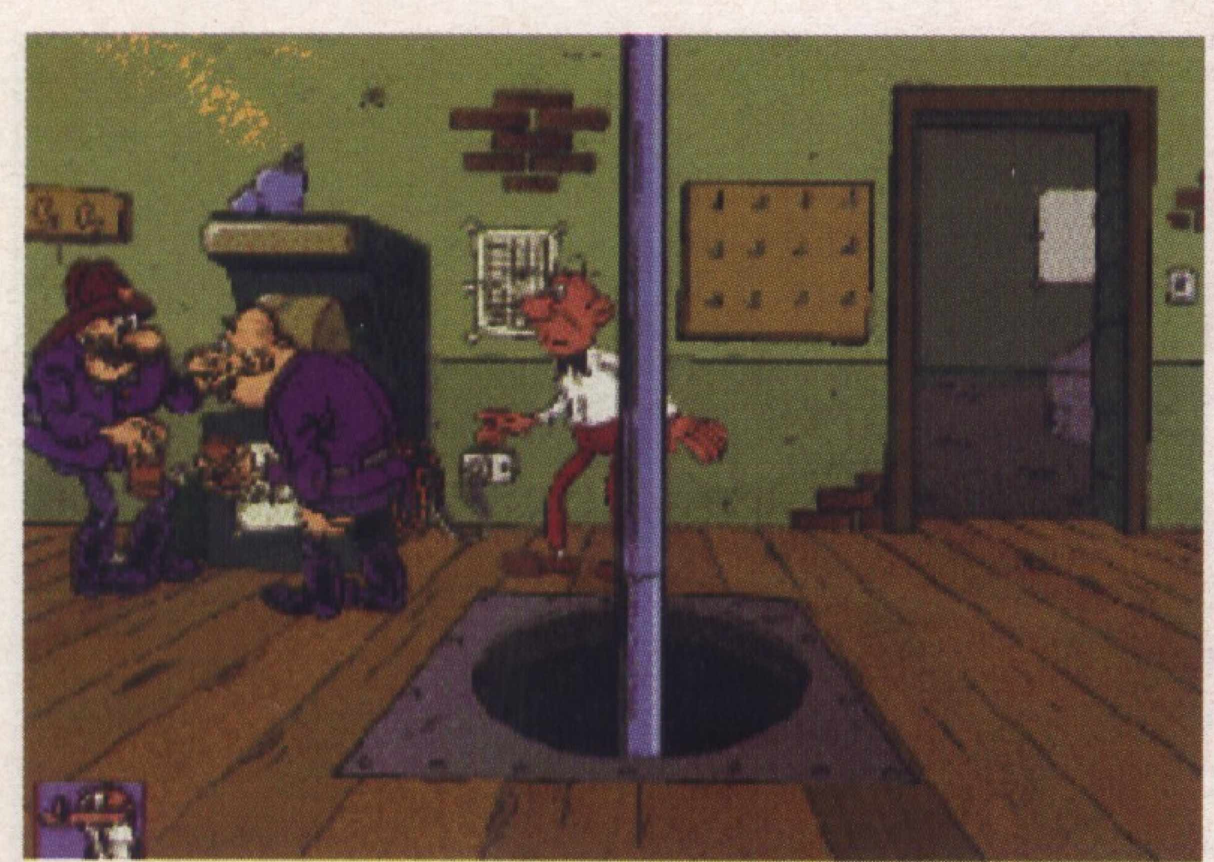
Para una mayor legibilidad en el código, y comodidad al programar, definiremos diez constantes, una para cada opción del menú (acción), más la acción especial "ninguna". Estos valores son especificados para cada acción del menú en la variable global *Columna[].Fila[].Opcion*. Estas constantes las llamaremos, de *o_Ir_a* (de opción ir a), *o_Hablar, o_Mover, o_Mirar, o_Coger, oDar, o_Abrir, o_Cerrar, o_Usar* y la acción especial ninguna *o_none.*

Por el momento, las llamadas a la función *sel_objeto* las dejaremos "comentadas" hasta que la implementemos. Las funciones *des_objeto* y *haz_accion* igualmente las trataremos más adelante.

La función *sel_accion* precisa de dos argumentos: *ident*, correspondiente al valor de la acción y *select* que, de valor verdadero, fijará la selección de la acción indicada en *ident*. El peso del *parser* estará en el bucle principal de nuestro juego. En este bucle principal situaremos el código para controlar el puntero del ratón y emular sus botones desde el teclado. Seguidamente detectaremos si el cursor del ratón se encuentra fuera del menú (las nueve celdas definidas). Esto ocurrirá si se encuentra a la izquierda de la primera opción, a la derecha de última o por encima de la primera.

Fragmento del *parser* correspondiente al bucle principal

```
IF ((mouse.x < Columna[0].x)  OR            // Cursor a la izquierda
     (mouse.x > Columna[2].xx) OR            // Cursor a la derecha
     (mouse.y <columna[0].fila[0].y))        // Cursor arriba de opciones
    // Si no opción seleccionada, bórrala
    IF (!accion.seleccionada) sel_accion(o_none, FALSE); END;
    IF (mouse.y <columna[0].fila[0].y)       // Cursor arriba de opciones
      // Si boton derecho, ANULA actual o Ir_a por defecto
      IF (mouse.right)
        IF (accion.id_obj_primario)          // Si hay actual -> anula
//          sel_objeto(o_none, FALSE);
        ELSE                                 // No hay actual -> Ir a
          sel_accion(o_Ir_a, FALSE);
        END;
      END;
    END;
  ELSE
    // Si dentro de columnas de opciones, ver cual es y si seleccionada o no
FOR (i=0; i<3; i++)
    IF ((mouse.x > Columna[i].x) AND (mouse.x < Columna[i].xx))
      FOR (j=0; j<3; j++)
        IF ((mouse.y > Columna[i].fila[j].y) AND (mouse.y < Columna[i].fila[j].yy))
          IF (mouse.left)
            sel_accion(Columna[i].fila[j].Opcion, TRUE);
          ELSE
            IF (!accion.seleccionada) sel_accion(Columna[i].fila[j].Opcion,
FALSE);END;
          END;
        END;
      END;
    END;
  END;
END;
```



Para hacer más fácil la programación se reducen al máximo las opciones siendo la opción "usar" la más elemental.



La interfaz puede ser muy variada, sacrificándose incluso en favor de los gráficos.



Los gráficos se han convertido en uno de los elementos más importantes del juego.

Si el cursor está fuera del menú de opciones, deberemos hacer:

- En el caso de que la acción actual no esté seleccionada, seleccionamos "ninguna", borrando así cualquiera que fuese la que estuviese en ese momento activa.
- Si nos encontramos sobre el menú y se pulsó el botón derecho del ratón: si tenemos seleccionado un objeto (al menos el objeto primario), seleccionamos objeto "ninguno", cancelando así cualquiera que fuese el objeto y la acción que estuviesen en ese momento activos. Si no había ningún objeto seleccionado (no existe objeto primario), seleccionamos la acción *Ir a* (nuestra acción por defecto).

Si por el contrario, nos encontramos dentro de los límites del menú, es seguro que el cursor estará en alguna de las nueve celdas, y por lo tanto, indica alguna de las nueve acciones posibles. Mediante un bucle *for* recorremos las columnas hasta encontrar la columna correcta. Encontrada la columna, con otro bucle *for* buscamos la fila. Localizadas la columna y la fila ya conocemos la acción, en el caso de no estar pulsado ningún botón, seleccionamos la acción con *sel_accion*, si el botón izquierdo del ratón estuviese pulsado, además de seleccionarla la fijamos.

Personalizando la interfaz

Ya tenemos nuestro menú de opciones completo (ver fichero *fuente_1.prg* del CD). Hemos pretendido acercarnos de una forma sencilla y rápida a cómo programar el menú, pero ya vimos al principio del artículo que la interfaz puede ser muy variada.

Podríamos hacer que el menú fuese sensible al puntero del ratón, destacando la opción sobre la que éste se encuentre. Añadiendo una simple función *repon_menu*, a la que invocaremos desde *sel_accion* podemos conseguir este efecto. Cada vez que seleccionemos una acción, *repon_menu* redibujará las opciones, comparando las coordenadas del ratón con la de cada opción, imprimiendo con una fuente distinta a aquella que corresponde a la posición del ratón. En el *Fuente_2.PRG* tenemos hechas las modificaciones para este efecto.

En el fichero *fuente_3.prg* del CD se encuentra la interfaz gráfica que usaremos a lo largo del curso. Para nuestra primera aventura en DIV, el menú de opciones será un gráfico que previamente habremos guardado. La función *pon_menu* consistirá simplemente en visualizar el gráfico del menú.

Rompiendo lo clásico

No nos cansaremos de reconocer que en una aventura gráfica lo primordial son los gráficos. No sólo no cuentan con un *parser* como las aventuras conversacionales, si no que ni siquiera el menú de opciones es importante. Hasta este punto se han buscado fórmulas para conseguir que los escenarios ocupen toda la pantalla, comiéndose el espacio del menú. Una solución muy elegante y socorrida es hacer del menú de opciones y del ratón un todo. En el fichero *fuente_4.prg* hemos preparado nuestro menú para funcionar así. Pulsando el botón derecho del ratón iremos cambiando la acción a realizar (modificándose el puntero del ratón), y con el izquierdo la ejecutamos. El gran inconveniente de este sistema es la necesidad de recorrer el menú de opciones secuencialmente, por lo que procura reducir en todo lo posible. Se evitan acciones como *abrir, cerrar* y *mover.* La acción *usar* suele ser directa, es decir, basta con elegir el objeto a usar para usarlo. Si el uso del objeto es combinado, al seleccionarlo aparecerá como puntero del ratón en espera de seleccionar el segundo objeto.

Otra solución muy elegante que encontramos en el fichero *fuente_5.prg* es la de ocultar el menú. El escenario ocupa toda la pantalla hasta que el ratón alcanza una determinada posición (intenta salir por la parte inferior de la pantalla). Es entonces cuando aparece el menú de opciones y de objetos. Cuando el ratón abandona la zona del menú éste vuelve a desaparecer.

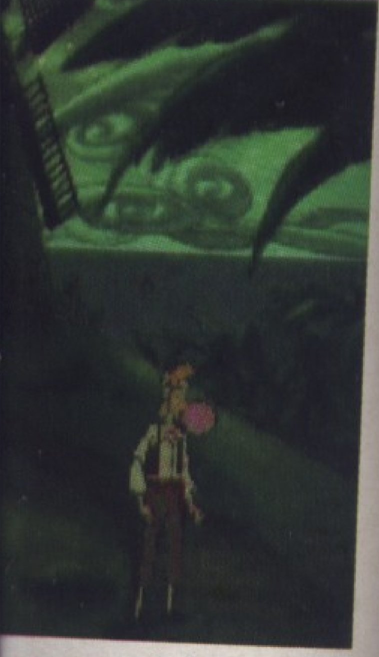
Miguel A. Barroso

Código de la función *repon_menu*

```
FOR (i=0; i<3; i++)
FOR (j=0; j<3; j++)
        marcado = FALSE;
        delete_text(columna[i].fila[j].id_t);
        if ((mouse.x >= columna[i].x) AND (mouse.x <= columna[i].xx))
          if ((mouse.y >= columna[i].fila[j].y) AND (mouse.y <= columna[i].fila[j].yy))
            columna[i].fila[j].id_t = write (f_blanco3,  columna[i].x+5,
                                    columna[i].fila[j].y+5,  0,  columna[i].fila[j].t);
            marcado = TRUE;
          END;
        END;
        if (!marcado)
          columna[i].fila[j].id_t = write (f_blanco,  columna[i].x+5,
                                   columna[i].fila[j].y+5, 0, columna[i].fila[j].t);
        END;
      END;
    END;
```

Nota sobre los ficheros fuente

Los ficheros fuente del CD están preparados para ser instalados en el directorio \div\cag\02 en la unidad donde tengas instalado DIV Games Studio



Programación de Juegos de Estrategia (II)



Un personaje carismático para un género cada vez más popular.

En este número añadiremos un par de detalles visuales a nuestro programa, pasando luego a seleccionar formatos de almacenamiento, datos de las unidades, etcétera. No veremos un avance considerable en nuestro juego, pero este paso nos permitirá programar mucho mas rápido en el futuro, evitándonos muchos problemas.

Lo primero, explicar los pequeños cambios de aspecto que ha sufrido el programa que empezamos a programar el pasado número. Con un pequeño vistazo vemos que el gráfico del ratón ya no es tan aparatoso, siendo ahora más manejable. Aparte, vemos que al mover el ratón, éste deja una estela de humo.

También podemos ver que a cada lado de la opción *Juego Nuevo* hay unas insignias que se oscurecen y reaparecen cíclicamente, y que cuando pulsamos las teclas *_up* o *_down*, suben o bajan entre las opciones del menú. Si pulsamos la tecla *Enter* se ejecutará la opción que tengamos seleccionada con las insignias.

Estos detalles tan solo le proporcionan más "movimiento" al menú, así como la opción de manejarlo por teclado, pero el método utilizado para generar la estela nos será de utilidad en el futuro, ya que los misiles de las unidades dejan una estela parecida.

La Estela del Ratón

El método utilizado para pintar la estela es muy sencillo:

Una función controla la posición del ratón (o misil) continuamente, y si este se separa de las últimas coordenadas guardadas más de una determinada distancia, ejecuta un proceso *humo()*, y actualiza las coordenadas de referencia.

El proceso *humo()* tan solo se dedica a pintar una animación del humo en las coordenadas que se le indican, tras lo cual "muere" (sale de la función). Este último proceso nos será útil también para las explosiones, ya que éstas no son más que una animación.

Suponiendo que cada fotograma lo ponemos en un *frame*, la animación durará tantos *frames* como fotogramas tenga la animación. Así, la estela del ratón no tendrá más de N segmentos, siendo N el número de fotogramas de la animación de humo (la función de la estela genera como mucho un segmento por *frame*).

Los detalles gráficos son de vital importancia.

Si queremos hacer la estela más larga, podemos aumentar el número de fotogramas de la animación o hacer que cada fotograma dure más de un *frame*. En nuestro programa cada fotograma de humo permanece 3 *frames* gracias a la orden *FRAME(300)*.

Las funciones de DIV para grabar y cargar datos en y de los ficheros sólo permiten manejar datos contiguos.

Array de fichas

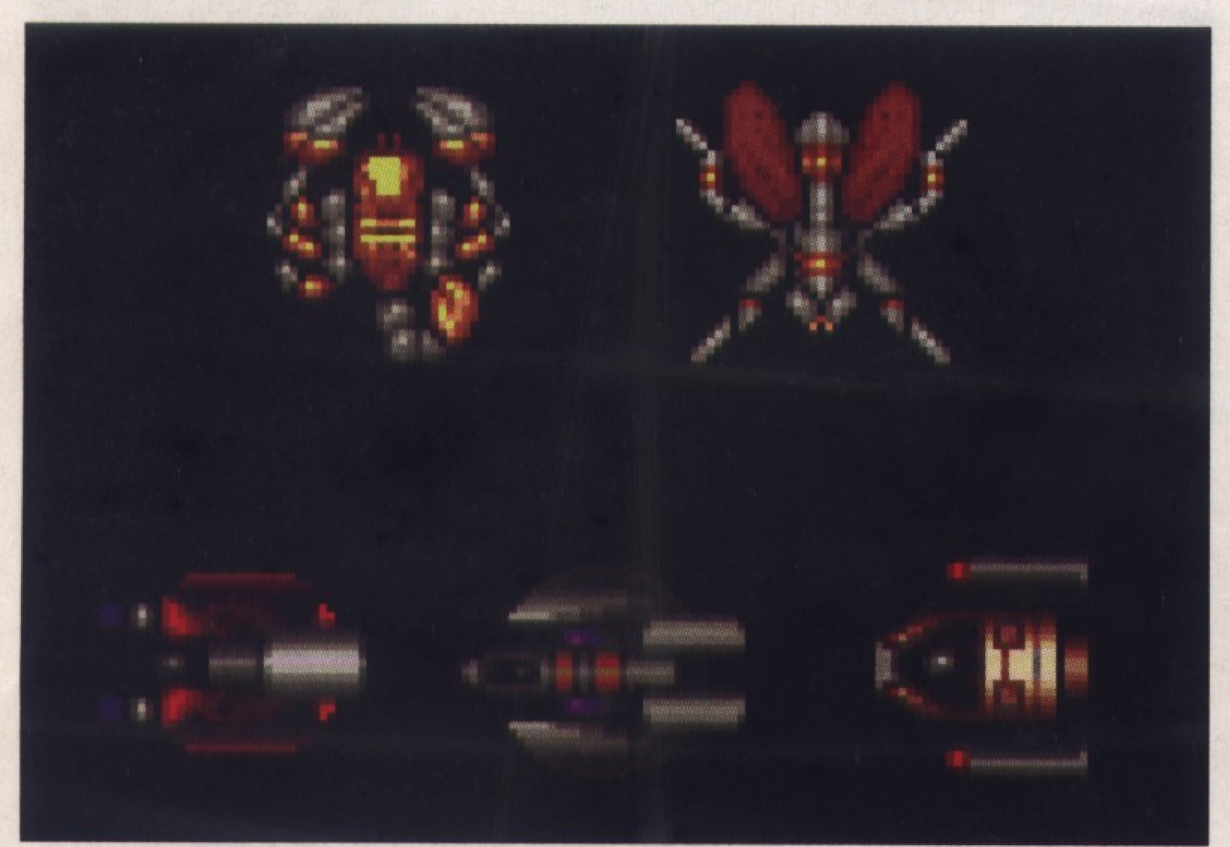
```
struct mobs[MAX_MOBS]
        ocupado;
        vida;
        tipo_unidad;
        bando;
        x; //solo para cuando se guarde
        y; //solo para cuando se guarde
        velocidad;
        destino_x; //cuadro de destino
        destino_y; //cuadro de destino
        accion;
end
```



Las unidades de nuestro juego

Las unidades de nuestro juego:

• **Obreros:**	Media en todo, poco daño en ataque.
• **Cazas ligeros:**	Poca armadura y poco daño, mucha velocidad y baratos.
• **Cazas pesados:**	Mucha armadura y mucho daño, lentos y caros.
• **Escorpiones mutantes:**	Intermedios entre caza ligero y caza pesado.
• **Moscas Gigantes:**	Como los cazas ligeros pero con capacidad de volar.



DIV posee unas excelentes librerías de gráficos.



Aquí se ve nuestro menú y la estela del ratón que manejamos.

Las Insignias Espaciales

El otro cambio importante en el aspecto del juego son las insignias, que permiten el uso del teclado para la selección de las opciones del menú. Este cambio no es tan simple como parece, ya que obliga a una estructuración de los menús que permite saber en qué coordenadas deben situarse las insignias.

Las insignias no son más que animaciones cíclicas oscilantes, es decir, que los fotogramas se repiten en orden inverso al llegar al último, y así continuamente. Por ejemplo, una animación cíclica oscilante de 4 fotogramas tendría una secuencia así: 1, 2, 3, 4, 3, 2, 1, 2, 3... hacer animaciones de esta forma nos ahorra todos los gráficos del recorrido descendente.

Los datos variables, como la vida, posición, estado, bando, etcétera, se almacenan en un *array* de estructuras global.

La dificultad que encierra la colocación de estas insignias estriba en cómo saber dónde está la siguiente opción por arriba o por debajo. El proceso que controla las animaciones de las insignias se llama a posteriori, modificando cuando se pulsa una tecla la posición de las insignias y la opción que representan. Si se pulsa la tecla *Enter* guarda en el contenido del puntero que se le pasa la opción sobre la que están las insignias y sale.

Como sabemos, en el menú que hicimos el pasado número la opción es igual a cero hasta que se pincha con el ratón en algún botón. Esto modifica el valor de la opción, provocando que se salga del bucle y se entre en un *switch-case*. Lo que hacemos en esta función es modificar el valor de opción, por lo que se sale del bucle de espera. Es un pequeño truco pero es efectivo al 100% y no es necesario modificar el código que ya teníamos. En cuanto a cómo saber dónde deben ir las insignias, lo hablamos ahora.

Elección de estándares

Esto es muy importante, ya que permite que una función sepa dónde buscar algo que ha hecho otra con sólo consultar un par de constantes. Debe ser objeto de estandarización lo siguiente:

- Unidades
- La organización de los gráficos entre *fpg's* (cuáles van en cada fpg).
- La colocación de los gráficos dentro de los *fpg's* (qué números tienen dentro de su fpg).
- Espacio entre opciones de cada menú.
- Para qué sirve cada tecla.
- Tamaño de la pantalla (modo): 320x200, 640x480, etcétera.

Todos estos datos se deben elegir al principio, y guardarlos en constantes si procede. El uso de constantes nos evita tener que buscar en todo el programa para cambiar el valor de uno de estos datos. Si guardamos las rutas de los ficheros que usemos en constantes, no tendremos problemas en el futuro si por cualquier razón se nos ocurre cambiar el nombre de un fichero.

Si no hubiéramos usado la constante, tendríamos que buscar cada referencia a ese fichero y cambiarla. Vista la importancia del uso de las constantes, vamos a por un caso más específico: nuestro juego.

Datos de la unidades

Como decíamos al final del pasado artículo, en este número vamos a seleccionar las unidades de cada bando, que para no tener que recurrir a un grafista, serán iguales en ambos bandos.

Los datos base de cada unidad se almacenan como constantes, que serán utilizadas por los procesos que necesiten la información.

Los datos variables, como la vida, posición, estado, bando, etcétera, se almacenan en un *array* de estructuras global. Algunos pensareis lo fácil que es definir todos estos datos como locales, y tenéis razón... siempre que no se tenga la intención de permitir grabar/cargar las partidas.

Las funciones de DIV para grabar y cargar en y de los ficheros sólo permiten manejar datos contiguos (cadenas de datos), y sólo se permite una grabación por fichero (siempre que se graba en un fichero, se "machaca" lo que había antes).

Así que creamos una estructura global que en nuestro juego se llamará *mobs*. En ella definiremos campos como la vida, la armadura, el bando, etcétera, y además un *flag* al que llamaremos "ocupado" que indicará si esa ficha está siendo usada para que las unidades puedan elegir una ficha libre.

Como local tan solo definimos una variable, que identifica la ficha que



La magia es uno de los elementos más atractivos a la hora de crear un videojuego.

Sobre los Gráficos

Hasta el momento los gráficos los hemos cogido de la librería de gráficos que trae DIV, pero a partir de ahora habrá que crear gráficos nuevos, ya que no nos valen los de este entorno. Por ejemplo, necesitamos gráficos para suelos de tamaño fijo, por ejemplo 32x32. Con ellos interconectados entre sí, formaremos un mapa grande sobre el que se moverán nuestras unidades. Si alguno desea mandar gráficos de muestra, sopesaremos la posibilidad de añadirlos al juego.



La creación de los personajes parte de diseños básicos.



Los personajes deben ser reconocibles a distancia.

contiene los datos de la unidad que está siendo representada por ese proceso (las variables locales existen en todos los procesos, pero cada proceso tiene las suyas), a la cual llamaremos *number*, ya que lo que guarda es el índice del *array* de estructuras donde esta nuestra ficha. Con todo definido, pasamos a la ejecución. Cada proceso unidad busca una ficha libre, tras lo cual la marca como ocupada poniendo el *flag* "ocupado" a 1 y guarda el índice en su variable *number*. Podrá entonces inicializar todos los datos de la ficha y ejecutarse normalmente. Cuando termine su ejecución, debe liberar su ficha poniendo el *flag* "ocupado" a 0 (no es necesario inicializarla entera, ya que lo hará el que la coja luego).

Características de las unidades

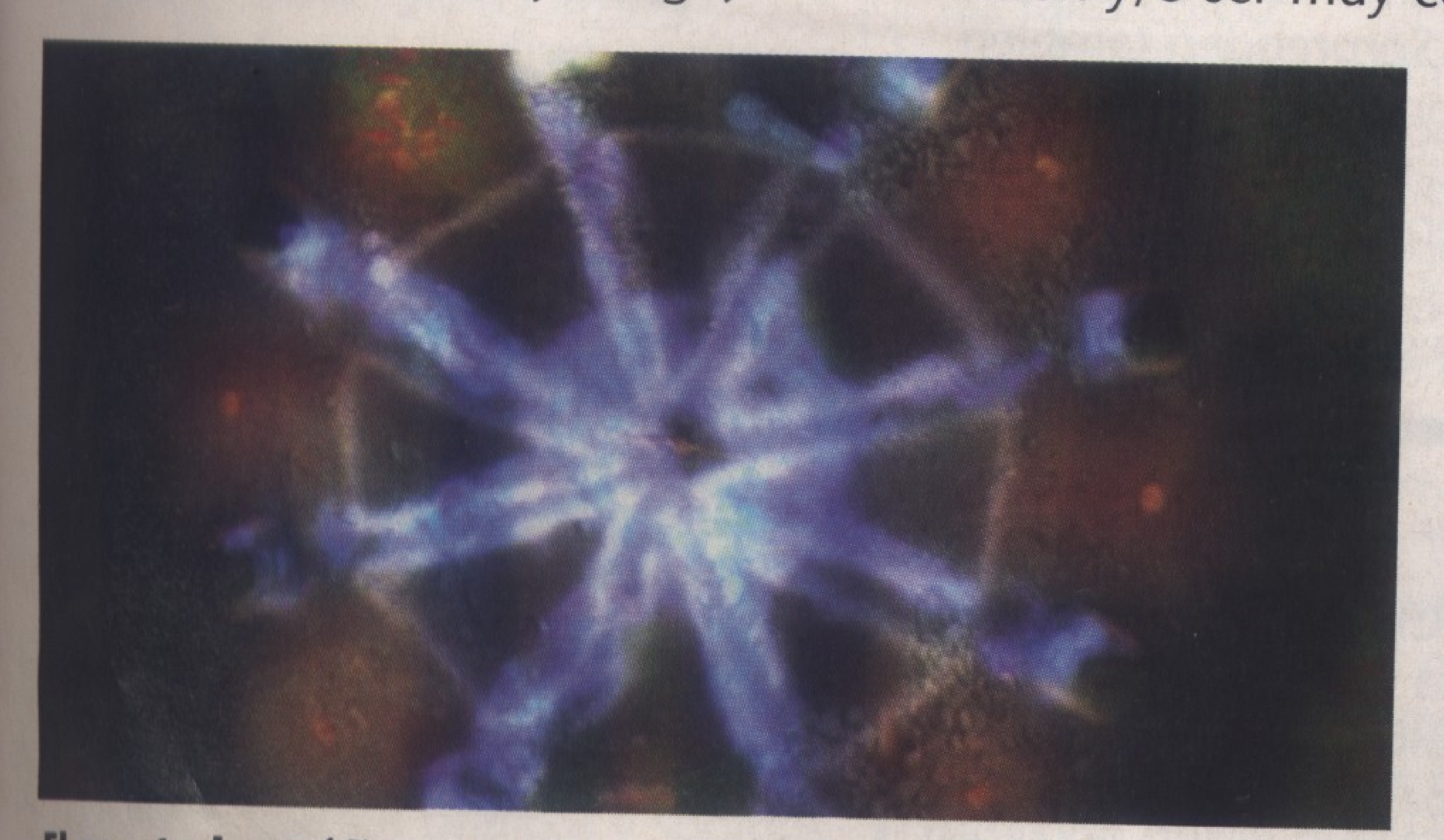
Una vez entendido esto, pasemos a elegir las unidades, y, lo que es más importante, sus características. Debemos tener en ambos bandos unidades ligeras y unidades pesadas. Para que las unidades de ambos bandos estén equilibradas, tenemos que elegir un método para repartir los puntos entre las características base, ya que si no se hace así, cierto tipo de unidades serán mejores en todo que otras. Todas las unidades deben tener algo mejor que las demás y algo peor. Por ejemplo las unidades rápidas no deben ser muy destructivas, y si lo son, deben ser muy lentas en su construcción y/o ser muy costosas.

Una vez elegidas las unidades y sus características, las definimos como constantes. Para hacer esto elegimos unos valores medios para cada característica, y los definimos. Luego para cada unidad ponemos un valor superior o inferior a la media de cada característica, controlando que la media final no se desequilibre demasiado.

En el próximo número

En el próximo número inicializaremos el *scroll*, explicaremos el *tileado*, y algunas cosillas más... Hasta el próximo número.

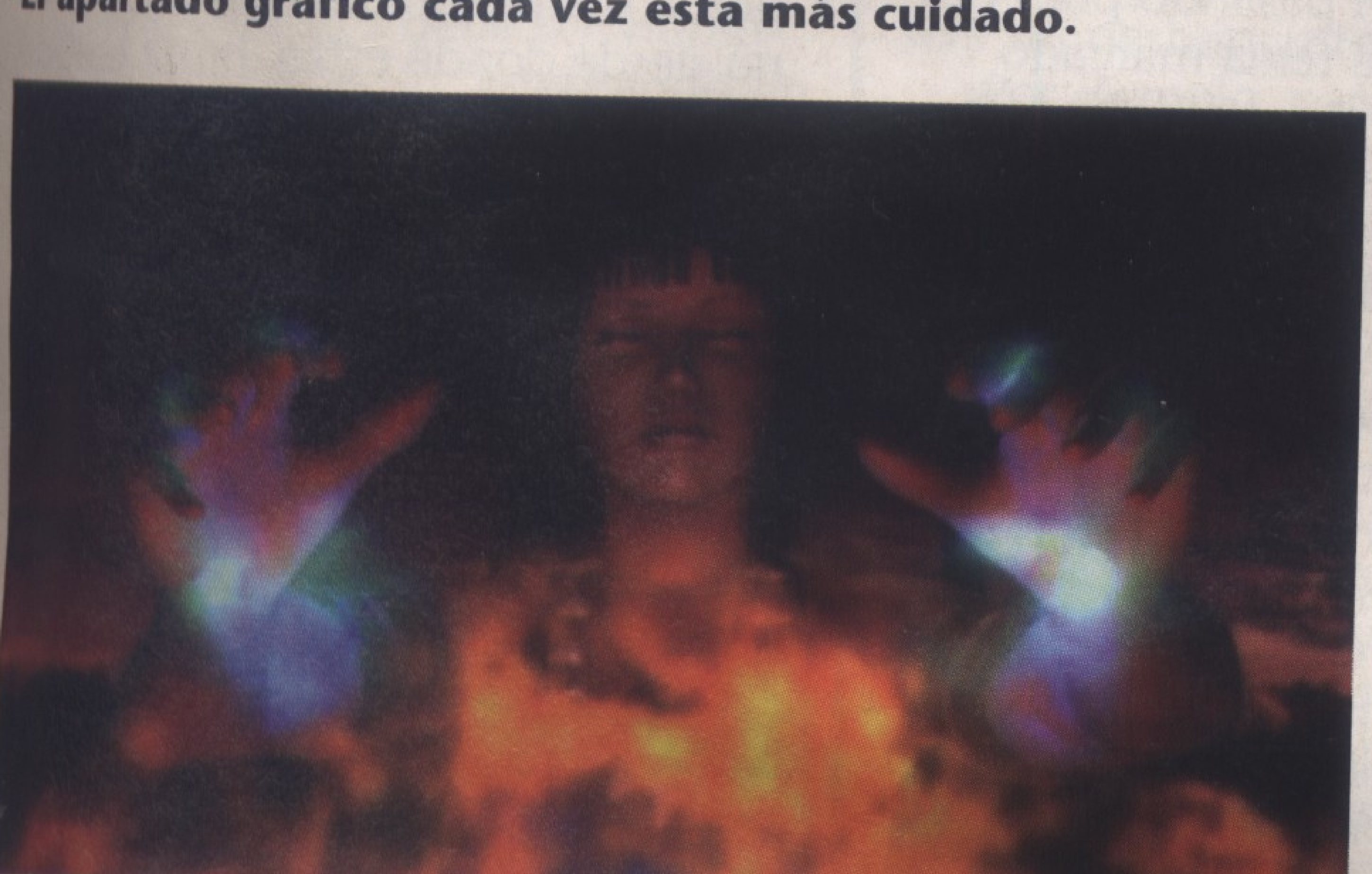
Emilio Llamas ALba



El apartado gráfico cada vez está más cuidado.



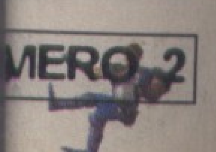
El atractivo de los personajes aumenta con el tiempo...



Un vídeo de introducción con encanto es imprescindible.



La presentación sienta las bases del desarrollo del juego.



El Francotirador

Dungeon Keeper

Hoy hablaremos de un juego no muy antiguo, pero que tampoco usa técnicas demasiado avanzadas como para considerarlo de última generación. Nos referimos a Dungeon Keeper (Guardián de la Mazmorra).



Un clásico del género que no envejece con el tiempo.

Empecemos el artículo comentando que el juego que nos ocupa podría haber sido hecho con DIV, ya que se basa en *sprites*. Lo único que no se podría haber realizado directamente con DIV serían los efectos de luz (que ya comentaremos más adelante) aunque podrían haberse simulado mediante transparencias.

El uso de *tiles* permite crear infinidad de mapas con sólo unos pocos gráficos. La iluminación de los *sprites* aumenta la sensación de volumen.

Gráficos

Comentábamos el pasado número que la familia *Warcraft*, debido al ángulo de vista que utiliza, debía usar distintos gráficos para cada orientación de los personajes. Pues en *Dungeon Keeper* el número de gráficos usados para las orientaciones aumenta, así como el número de bichos.

Además, permite vista en primera persona, lo que obliga a tener otro juego de gráficos distintos para cada monstruo (el ángulo es diferente).

Lo primero que llama la atención es la iluminación, que afecta a los *sprites* dándoles más brillo por las zonas iluminadas, aumenta la sensación de volumen. Sorprenden también las sombras proyectadas por los personajes cuando son iluminados, ya que cubren casi todos los ángulos posibles, y no son un borrón sino que muestran la postura que en ese momento tiene su "dueño". También en el artículo anterior comentábamos el uso de una matriz como soporte del mapeado. En este juego puede verse esta técnica en muchos detalles, por ejemplo, el manejo del suelo (construir, cavar). En cambio, en cada cuadrado puede haber más de un personaje (a diferencia de la familia *Warcraft*). Esta cuestión quizá obligue a usar colisión por distancias como la que se debe usar en el modo-7.

Inteligencia Artificial

El algoritmo de búsqueda de caminos que aparentemente utiliza la familia *Warcraft* es muy simple, pero aquí no tendremos tanta suerte: los personajes están haciendo cosas aunque nosotros no se lo digamos, y si nos fijamos vemos que todos los bichos saben dónde está la sala del tesoro, el criadero o el templo más cercano. Esta información está directamente en la matriz que comentábamos antes. Para moverse por la matriz, precalculan un camino desde donde están, hasta el punto destino. Para calcular este camino existen múltiples algoritmos, por lo que es imposible saber a ciencia cierta cuál es el utilizado. Aun así, podemos ver un algoritmo bastante conocido en el cuadro.

Despedida

Nos despedimos hasta otra comentando que todo lo que vayamos viendo en el francotirador lo usaremos para la programación del juego del curso en el futuro. Nos leemos...

Emilio Llamas Alb...

Pseudocódigo de Exploración de Grafos.

Pseudocódigo ExploraGrafos:

- Guardar en una cola (ABIERTOS) el punto de origen (S).
- Inicializar a vacía otra cola (CERRADOS).
- Si ABIERTOS está vacía, algo ha ido mal... no se pudo encontrar un camino.
- Sacamos el primer elemento de ABIERTOS y lo guardamos en una variable N y en CERRADOS.
- Si N es un destino, entonces calculamos el camino gracias a los apuntadores que unen N con S (el origen).
- "Expandimos" N, es decir, metemos todas las casillas adyacentes a N (y accesibles) en X, siempre y cuando estas casillas no fueran ascendientes de N (tuvieran un apuntador desde N hasta ellas).
- Poner un apuntador a N desde los "sucesores" de N que no estén ni en ABIERTOS ni en CERRADOS. Y para todo H contenida en X:
  - Si H está en ABIERTOS o en CERRADOS, comprobamos si el camino que lleva hasta H pasando por N es mejor que el que tenía marcado hasta ese momento, en cuyo caso, modificaremos el apuntador que tenía H para que apunte a N.
  - Si H está en CERRADOS, decidir para los descendientes de H si el camino a través de H es mejor que el que tenían hasta ese momento, en cuyo caso, modificaremos los apuntadores de los descendientes de H.
- Reordenar ABIERTOS de acuerdo a una heurística (primeros los que más cerca estén en línea recta al destino, y últimos los que mas lejos estén).
- Ir a 3).

Cómo programar un juego de rol (I)

Busca el Elfo que llevas dentro

Después de haber visto en el primer número de la revista alguna de las principales características de los juegos de este género, y qué era necesario para poder realizar uno, a partir de este número iniciamos un cursillo que nos introducirá en un mundo fascinante de la mano de un experto del género.

Antes de empezar a programar un juego de rol (RPG : Role Playing Game) tendremos presente toda una serie de aspectos relacionados con el diseño del juego:

Primera etapa

En esta etapa de desarrollo del juego definiremos la estética y el argumento del juego. Esto es, probablemente, una de las partes más importantes del juego y, por tanto, debe realizarse minuciosamente, ya que en un RPG suele ser mucho más importante el argumento y la originalidad del juego que sus gráficos y sonidos.

El argumento debe ser vibrante, rápido e intenso, intentando no aburrir nunca al jugador. Para conseguir un buen argumento debemos planearlo con sumo detalle, es decir, debemos hacernos preguntas tales como:

¿Dónde transcurrirá el juego? ¿Cómo será el protagonista? ¿Y sus compañeros? ¿Cuál será la causa que impulse al protagonista a luchar? ¿Quién será el malo? ¿Habrá alguna desgracia que impedir? ¿Y las mazmorras y ciudades? ¿Será futurista o ocurrirá en el pasado?

Para responder a todas estas preguntas, y otras que irán surgiendo, pensaremos con calma, tratando de resolverlas una a una, ya que si intentamos responderlas todas de golpe nuestro proyecto será un desbarajuste.

Por ejemplo, para definir el mundo donde se desarrolla el juego, se pueden escribir pequeños textos en los que explicaremos fragmentos de acontecimientos pasados que sitúen al jugador. Para determinar la ambientación y el diseño de las ciudades podemos hacer dibujos esquemáticos o simples descripciones de cómo son estas ciudades.

El objetivo a conseguir en el juego es también importante, dado que es una de las cosas que impulsarán al jugador a jugar con más o menos intensidad. El objetivo del juego puede ser muy variopinto, pero cuanto más original sea, mejor. En muchos casos, para finalizar el juego debemos salvar el mundo o matar al "malo". Este último será, junto al protagonista, el personaje mejor trazado, dada su importancia en el desarrollo del juego.

Un aspecto fundamental de cualquier juego de rol es el protagonista, como ya hemos dicho, pero no debemos olvidar a sus amigos. Este apartado debe estar muy cuidado ya que el jugador debe sentirse identificado con el personaje y sus compañeros, consiguiendo así una mayor motivación que impulsará al jugador a dedicarle más horas al juego.

Los rasgos principales de un personaje principal en un juego de rol suelen ser:

Cuadro

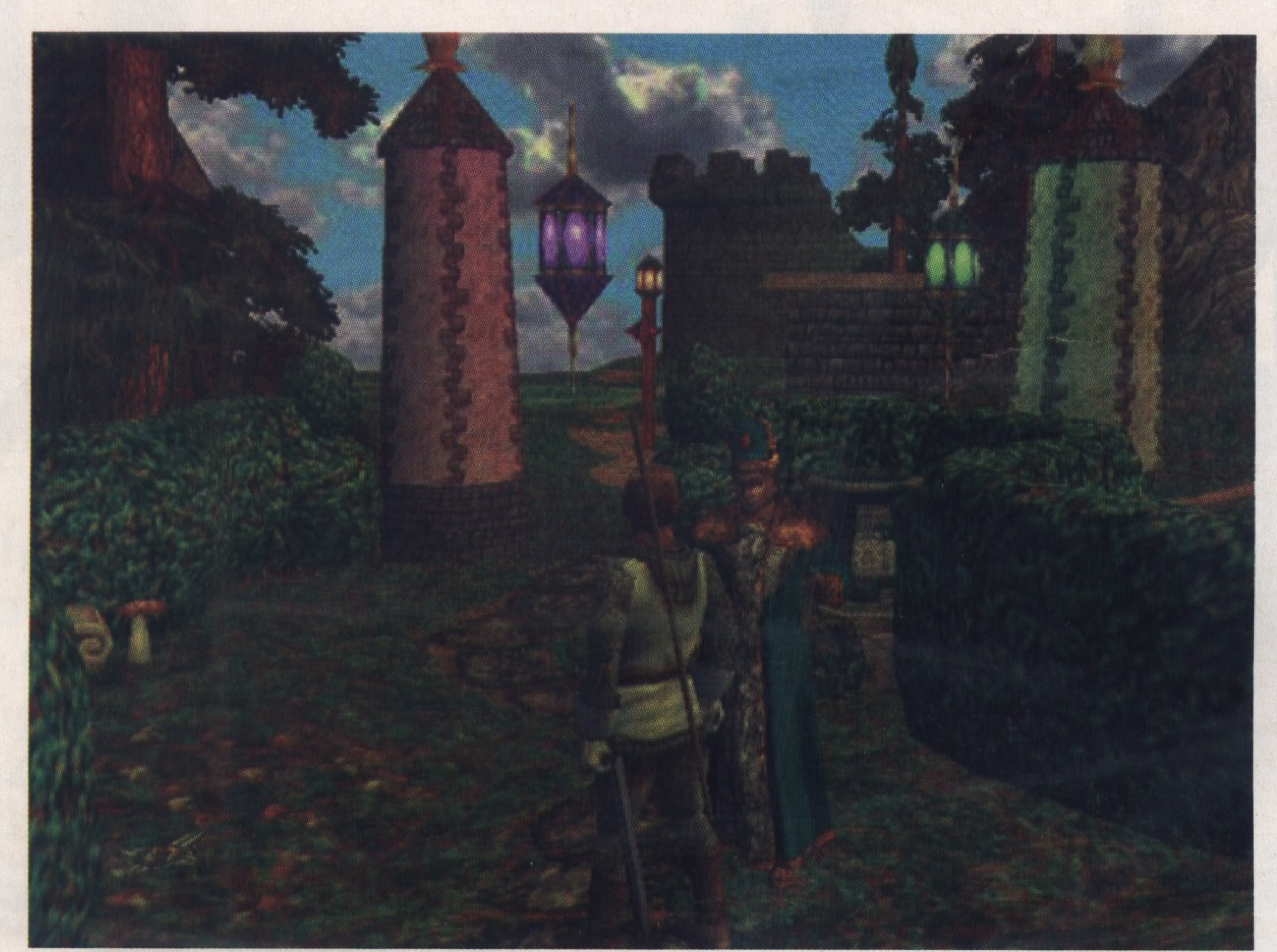
Razas
- Humano
- Elfo
- Medio-Elfo
- Enano
- Etc.

Clases
- Luchador
- Mago
- Clérigo
- Ladrón
- Paladín
- Etc.

Personalidades
- Legal-Bueno
- Caótico-Bueno
- Neutral-Bueno
- Legal-Neutral
- Caótico-Neutral
- Neutral-Neutral
- Legal-Malo
- Caótico-Malo
- Neutral-Malo
- Etc.

Atributos
- Fuerza
- Inteligencia
- Destreza
- Esperanza
- Carisma
- Nivel
- HP: Hit Points (Puntos de Vida)
- MP: Magic Points (Puntos de Magia)
- Experiencia
- Etc.

Moderno RPG en 3D.


Un entorno excepcional para Ultima Ascensión.

Muchos juegos nos permiten elegir las características que nuestro personaje tendrá, por lo que sería una buena idea apuntar todo lo que se nos ocurra sobre cómo serán los personajes en nuestro RPG.

A simple vista, puede parecer que el hecho de la elección del personaje es un acto banal e inútil. Pero si nos fijamos, veremos que es una de las cosas más importantes y vitales para empezar la partida con buen pie. La incorrecta elección de nuestro personaje podrá acarrear graves problemas durante el juego, pero si por el contrario, elegimos un buen personaje, conseguiremos muchos más éxitos durante el juego. En un juego de reciente aparición, *Baldur's Gate*, se da mucha importancia a este aspecto, hasta tal punto que podemos llegar a tardar media hora en crear nuestro personaje.

En un RPG, el argumento ha de ser vibrante para que nunca aburra y te ha de prestar especial atención a la elaboración de los personajes

Segunda etapa

Una vez que has pensado en todo lo relacionado con el juego, es una buena idea clasificarlo en diversos apartados para facilitar el diseño:

- **Lugares (o habitaciones)**: Es decir, pueblos, cuevas, mazmorras, etc. También deberás incluir en este apartado todos los lugares que el personaje puede visitar durante el juego, junto con una breve descripción que ayudará a tenerlo todo más ordenado.
- **Mapa**: Deberás crear un mapa con todos los lugares citados en el apartado anterior.
- **Objetos**: Aquí se tendrán que apuntar todos los objetos que habrá en el juego, junto a una breve descripción de sus propiedades. En esta lista se incluirán objetos relacionados con el armamento que usarán, tanto el personaje principal como otros personajes.
En este mismo apartado, se podrá elaborar un subapartado en el que se hablará de aquellos objetos que son necesarios para avanzar en el juego o para conseguir objetivos específicos.
- **Habitantes del juego**: Este apartado lo podemos dividir en 3 subapartados:
  - **NPC** (Non Player Characters): Aquí se incluirán todos aquellos personajes que aparentan tener inteligencia propia, pero que no son controlados por el jugador. En este subapartado incluiremos a los "jefes", porque estos luchan de una manera diferente al resto de los enemigos del juego.
  - **Personajes controlados por el jugador**: Sin duda alguna los más importantes. Evidentemente, aquí incluiremos al protagonista y a todos sus compañeros, controlados por el jugador. Una manera de hacerlo todo más sencillo es redactar un pequeño informe sobre cada personaje, mencionando su pasado, atributos y cualquier otra peculiaridad que posea.
  - **Enemigos**: Esta es una sección importante. Para tener el conjunto más organizado haremos una lista con todos ellos. En dicha lista indicaremos dónde aparecen y qué características tienen. Los enemigos sólo atacarán al jugador y de una manera bastante rutinaria, basándose en unos patrones que nosotros designaremos.
- **Cambios durante el juego**: En este apartado apuntamos todas aquellas cosas que cambiarán durante el juego. Ejemplo: En *Zelda,* de Super Nintendo, empiezas a jugar en un mundo bonito y colorido llamado Mundo de la Luz, pero llega un momento en que tienes que ir al Mundo de la Oscuridad, de características antagónicas al de la Luz.

Pero, no solamente mencionaremos aquí cambios de mundo,



La tercera parte de uno de los títulos señeros del género.


Los efectos especiales cada vez están más logrados.

sino que también apuntaremos cualquier cambio en la personalidad de los personajes o en los atributos de un objeto.

- **Texto**: En esta extensa sección escribiremos los textos que cualquier personaje del juego pueda decir. Los diálogos en los juegos de rol son muy importantes, ayudan al personaje a introducirse y a interactuar con el mundo imaginario del juego. Mucha gente, cuando ve demasiado diálogo se "marea" y pierde interés por el juego, así que deben ser fluidos para que no resulten pesados al jugador. Si los textos de un personaje cambian según lo que ocurre, lo puedes apuntar en la sección de cambios o en esta misma.
- **Acciones**: Escribiremos en este apartado todas las acciones que se realizan en el juego o que se deben realizar para conseguir algún objetivo en especial. Cualquier acción que ocurra, por muy puntual e irrelevante que sea para el juego, debe ser detalladamente explicada aquí. Ejemplo: Para entrar en la 3ª mazmorra necesitas tener una llave y haber acabado la 2º mazmorra previamente.

Tercera Etapa

Una vez llegado a este punto, deberías saber cómo empezar a programar el juego.

Es una buena idea hacer una lista con enlaces de todos los objetos del juego. Esta lista tendrá un enlace que ordene los objetos dentro de una jerarquía, y un segundo enlace que los mantenga ordenados entre las distintas jerarquías. Esto último parece complicado (aunque no lo sea), así que a continuación pongo un ejemplo:

- Lista: Objetos que hay en la nevera (zumo, leche, etc.). Con esta primera lista mantenemos ordenados todos los objetos que hay en la nevera.
- Lista: La leche está DENTRO de la nevera. Mediante esta segunda lista indicamos que el objeto leche está dentro de la nevera.

Todos los objetos tendrán un único código de identificación. Deberás tener una lista con los números y a qué objeto se refiere cada número. Además de tener un único código de identificación, todos los objetos deberán guardar datos sobre lo que pueden hacer en el juego. Otro aspecto importante aquí es dar los atributos a los habitantes de nuestro juego:


El medievo sigue siendo la época preferida del rol.

- **Enemigos**: Indicaremos su manera de luchar, cuándo aparecen y con qué frecuencia, vida (HP), nivel (LEVEL), experiencia (EXP), puntos de magia (MP), etc.
- **NPC**: Estos personajes, de gran importancia para el juego, deben valerse por sí mismos dado que no son controlables por el jugador. Debemos procurarles una manera de comportarse para cada momento del juego.

En todo juego de rol, cada criatura ha de tener un inventario, especialmente los personajes controlables, pero también los enemigos para que, cuando sean eliminados, donen sus pertenencias al jugador. El inventario es donde se guardan todos los datos relacionados a los objetos y pertenencias que cualquier personaje posee.

También declararemos variables para todos los textos que aparezcan en el juego. A partir de este momento es cuando realmente empieza la programación del juego...

Cuarta Etapa

En esta etapa empezaremos a crear el código del juego. Podemos empezar por ejemplo, por la interfaz, el sistema de combate, el sistema de comercio o las rutinas gráficas.

- **Sistema de comercio**: El sistema de comercio se encarga de describir cómo las criaturas animadas (NPCs y jugadores) pueden comerciar. Este sistema de comercio se puede basar en el intercambio, en la compra/venta o en cualquier otro método que se te ocurra.
- **Sistema de combate**: El sistema de combate es un conjunto de reglas que se encargan de definir cómo combatirán los enemigos de nuestro juego.

Muchas veces se sacrifican los gráficos y el sonido del juego en favor de un buen argumento

- **Rutinas gráficas**: Las rutinas gráficas son el conjunto de instrucciones que hacen que las acciones de cualquier objeto sean mostradas al jugador por medio del monitor.

En este momento es cuando empieza la etapa de testeo y corrección de errores (bugs). Para facilitar esta labor, lo mejor es testear las diversas partes del programa separadamente. Cualquier error que encontremos lo apuntaremos, para corregirlo posteriormente. Una vez lo hemos revisado todo, sólo faltan pequeños detalles que mejorar antes de dar por finalizado el juego.

Maqnaziller

Fasttracker 2.08

La banda sonora de tus juegos

Hoy por hoy, la música es imprescindible para que se pueda crear la mejor ambientación dentro de un videojuego, haciéndolo todavía más atractivo e interesante. En esta sección aprenderemos el manejo de los programas que pueden utilizarse para la confección del sonido.

En el número anterior de la revista tratamos acerca del manejo de un programa de edición de *samples* como herramienta en la generación de efectos sonoros para nuestros propios juegos. Este programa podía utilizarse para *samplear* música creada por otros medios e incluso para generar melodías sencillas, pero su utilidad para componer nuestra propia música era demasiado limitada. Podría ser socorrido hacer uso de la creación de estudio de alguno de nuestros intérpretes favoritos como banda sonora de uno de nuestros juegos, pero esto mermaría la originalidad del producto final. Los videojuegos más conocidos se caracterizan por tener su propia música, compuesta pensando en los mismos, y muchas de estas composiciones han sido creadas con secuenciadores como el que vamos a tratar en el pequeño curso que comenzaremos en este número y que continuará durante algunos meses.

Rápidamente, somos capaces de construir una base rítmica sobre la que desarrollar nuestra composición

Procuraremos no enfocarlo como un manual de usuario del programa en sí, sino como una especie de guía para componer canciones de principio a fin, dando incluso algunos conceptos básicos de armonía y solfeo.

¿Qué es un secuenciador?

Básicamente, un secuenciador es una herramienta de composición musical que permite escribir una partitura con múltiples instrumentos y hacer a su vez que ésta se escuche, de manera que podemos ir escuchando "in situ" aquello que vamos creando. Esto último es lo que hace que los secuenciadores sean muy útiles para músicos amateur, ya que ofrecen la posibilidad de componer canciones complejas sin necesidad de poseer conocimientos rigurosos de armonía o solfeo, aunque éstos resultarían, sin duda, bastante útiles.

Hay dos tipos de secuenciadores: los que funcionan vía MIDI, o los que, como Fasttracker, funcionan mediante *samples*. Esto significa que los instrumentos que interpretarán nuestras composiciones los adquiriremos sampleándolos con nuestra tarjeta de sonido, o tomándolos de las ricas librerías de *samples* ya existentes. Este tipo de secuenciador resulta más versátil, ya que ofrece la posibilidad de incluir sonidos definidos por el usuario, tales como voces humanas o secuencias complejas de instrumentos musicales, como por ejemplo un solo de guitarra completo.

La interfaz de fasttracker

En la pantalla de Fasttracker encontramos los siguientes elementos:

- *Tabla de patterns*: ubicada en la esquina superior izquierda. Un *pattern* es un segmento de la canción, de mayor o menor longitud, que podrá repetirse tantas veces como queramos. En esta tabla será donde daremos orden a los *patterns* creados por nosotros mismos, de manera que esto será lo que defina la estructura de la canción. Al lado derecho de esta tabla se encuentra el lugar donde podremos seleccionar la velocidad de reproducción de la canción y el número de líneas que compondrá cada *pattern*.
- *Banco de instrumentos*: se encuentra en la esquina superior derecha de la pantalla. Es en esta parte donde aparece la totalidad de los instrumentos de la canción. El activo en cada caso será aquel que esté resaltado en un color más claro. Podremos cambiar de banco pulsando uno de los ocho botones del extremo izquierdo, teniendo ocho instrumentos por banco, de manera que se dispondrán como máximo de un total de 64 instrumentos por canción.
- *Barra de botones*: posicionada entre los dos elementos anteriormente citados. Desde ella podremos activar las numerosas opciones de Fasttracker.
- *Conjunto de gráficas de onda de cada canal*: situada bajo la tabla de *patterns*. Cuando reproducimos una canción observamos en esta sección el estado de la onda en cada canal (*track*) y en cada momento del tiempo.
- *Partitura*: ocupa la mitad inferior de la pantalla y es donde escribiremos las notas de nuestra composición.

Empezando por los cimientos

Ahora que ya nos hemos familiarizado con la pantalla pasaremos a la parte práctica. En lugar de teorizar extensamente con las magnificencias del programa, vamos a empezar a preparar lo necesario



Con Fasttracker podremos emular a las estrellas del rock.

para realizar nuestra primera composición musical.

En primer lugar pincharemos con el ratón en la opción *Disk op.* de la *Barra de botones*. En la parte superior izquierda aparece ahora la palabra *item*, que define el tipo de elemento que vamos a cargar o salvar: una canción entera (*Module*), un instrumento (*Instr*), un instrumento sin información adicional (*Sample*), una parte de canción (*Pattern*) o una sola pista (*Track*). En este caso elegiremos *Instr.* Ahora buscaremos en alguno de nuestros dispositivos de almacenamiento un sonido de bombo de batería. Seleccionamos el lugar en el banco de instrumentos donde lo queremos colocar (lo colocaremos en el primer banco, en la primera posición) y después hacemos doble clic sobre el archivo de extensión XI que representa nuestro sonido de bombo.

Ya tenemos cargado un instrumento y por supuesto podemos hacerlo sonar haciendo uso del teclado de nuestro ordenador. En Fasttracker las teclas pueden ser utilizadas como si de un piano se tratase. Pulsamos por ejemplo la tecla <Q>, que suele ser la que hace sonar el instrumento en su octava natural, y comprobamos que suene el bombo. Ahora queremos establecer una base rítmica percusiva, de manera que pulsamos la barra espaciadora, que es la que activa el modo que nos permite escribir en nuestra partitura. Colocamos el cursor en el canal (*track*) 0, línea 0 y pulsamos la tecla <Q>. Vemos que aparece escrito el siguiente código: C-5 1000. Las notas se escriben en notación americana, por eso aparece la C, que corresponde a la nota DO. El 5 representa la octava de la nota (la octava es la que determina la frecuencia de la nota, es decir, a octavas menores la nota será más grave y a octavas mayores más aguda). El 1 significa la posición que ocupa el instrumento en el banco de instrumentos y los tres últimos dígitos representan el efecto aplicado a la nota en cada caso, pero la cuestión de los efectos es bastante más compleja y la trataremos mucho más adelante.

Para completar la base rítmica escribiremos la misma nota y en el mismo canal repetidas veces con una separación de 8 líneas, es decir, en la 8, en la 16, en la 24, en la 32, 40, 48, 56 y 64 (los *patterns* tienen 64 líneas por defecto, que es lo más indicado para realizar ritmos 4x4).

Pulsamos *Play ptn.* en la *Barra de botones* y escuchamos el resultado. Ya tenemos sonando un metrónomo sobre el que construiremos nuestra melodía. Cuando queramos detener la audición bastará con pulsar una vez la barra espaciadora.



Reclutando más músicos

Repetimos el proceso para cargar un instrumento, pero esta vez lo haremos en la segunda posición del primer banco y lo que cargaremos será un bajo, que será el que sirva de puente entre la base rítmica y el resto de los instrumentos. Escribimos esta vez en la pista 1 de la partitura, dejando sólo una línea de separación entre nota y nota y comenzando en la línea 0. Una buena manera de conseguir que la música que compongamos no sea disonante sería escribir siempre con notas blancas (sin tonos sostenidos => #) y comenzar siempre los compases con las notas DO (teclas Z,Q,I) o LA (teclas N,Y), de manera que nos aseguremos de estar en la tonalidad de Do Mayor - La menor. Por supuesto, este es un truco para utilizar en nuestras pruebas y con el fin de poder improvisar en nuestros primeros pasos, pero después no debemos limitar nuestra creatividad a una sola tonalidad, sino dejar que la improvisación y nuestro oído actúen por sí solos.

Una vez que estemos satisfechos con la base rítmica que hemos creado, cargamos un nuevo instrumento. Ahora sería interesante habilitar uno de los canales para improvisación. Para ello nos dirigimos con el ratón al conjunto de gráficas de onda de cada canal, que se encuentra debajo de la tabla de *patterns*. Pulsando sobre cada gráfica con el botón izquierdo del ratón podemos silenciar (*mute*) los canales y con el botón derecho habilitamos o deshabilitamos cada canal para improvisación. Haremos esto último sobre los canales en los que queramos improvisar, de manera que aparezca la palabra *rec.* en ellos. Reproducimos el *pattern* mientras que improvisamos tocando el teclado de nuestro ordenador sobre la base rítmica. Si hemos utilizado el truco enunciado antes, siempre nos sonará más o menos bien si tocamos siempre notas blancas (Do, Re, Mi, Fa, Sol, La, Si). Cuando hayamos conseguido alguna melodía que nos satisfaga podemos intentar escribirla mediante el método convencional, que a no ser que estemos muy familiarizados con él nos resultará muy difícil; o bien podemos escribir nuestra melodía en directo, es decir, mientras suena la música, haciendo uso de la opción de la barra de botones *Rec. ptn.*, que pondrá en movi-

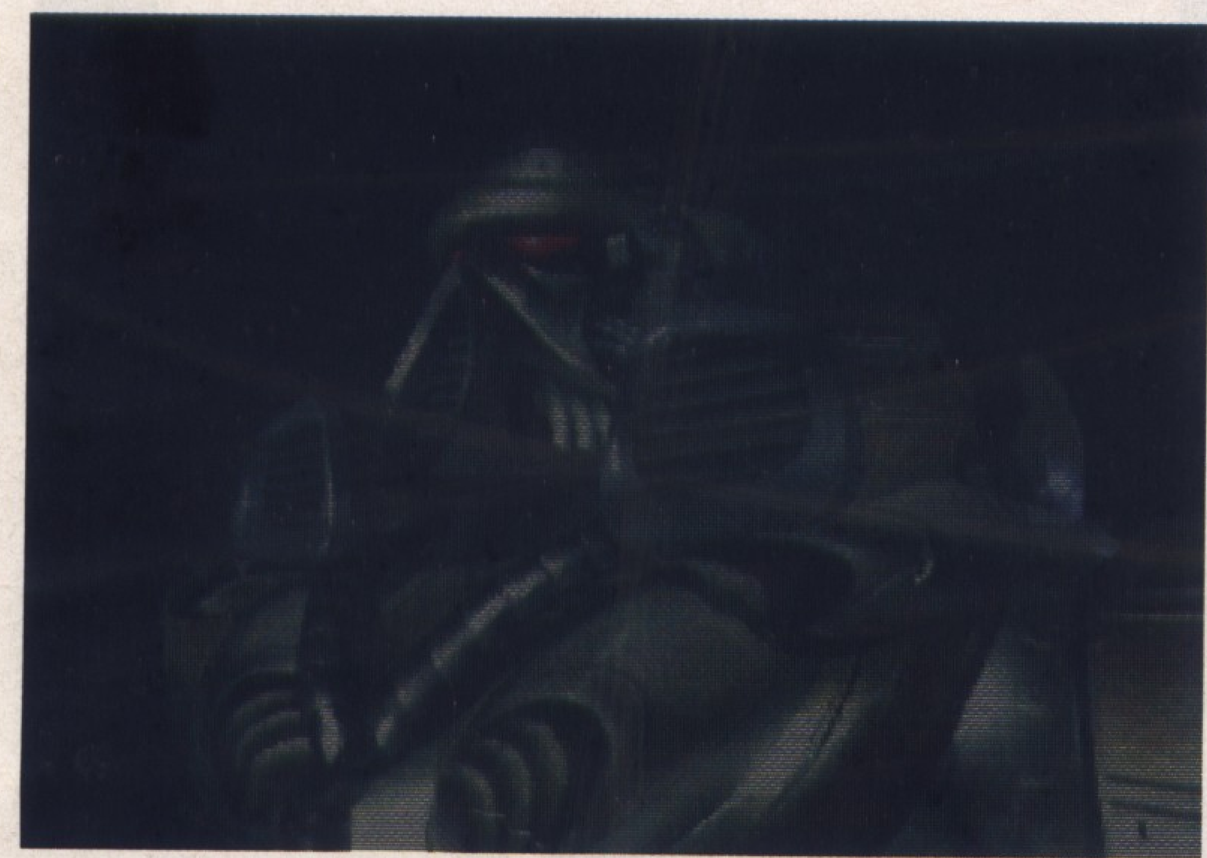
Disposición de las notas en el teclado

OCTAVA AGUDA:

DO# RE# FA# SOL# LA#
2 3 5 6 7
DO RE MI FA SOL LA SI DO
Q W E R T Y U I

OCTAVA GRAVE:

DO# RE# FA# SOL# LA#
S D G H J
DO RE MI FA SOL LA SI DO
Z X C V B N M ;



La tétrica banda sonora del juego Terracide es un buen ejemplo de lo importante que puede ser la música para lograr una buena ambientación.

miento la partitura, haciéndola sonar y permitiéndonos a su vez escribir sobre ella.

De camino hacia una sinfonía

Con lo que hemos visto debemos ser capaces de componer un *pattern* con diversidad de instrumentos, pero necesitaremos más de un *pattern* para completar una canción si no queremos que ésta resulte extremadamente tediosa.

Con un secuenciador como Fasttracker no sólo se puede crear música electrónica, sino que se pueden abarcar todos los estilos existentes

Al lado de nuestra tabla de *patterns*, que ahora mismo deberá constar de un único elemento (el *pattern* 00 en la posición 00), observamos los botones *Ins* y *Del*, cuyas funciones son las de insertar nuevos *patterns* en la lista o eliminarlos. Pinchamos en *Ins* para añadir un nuevo elemento, de manera que ahora tenemos dos veces el *pattern* 00. Si queremos que la segunda porción de nuestra canción no sea una repetición de la primera, tendremos que utilizar las flechas que se encuentran entre los botones *Ins* y *Del* sobre la posición deseada de la lista de *patterns*, para indicar el *pattern* que queremos que se reproduzca en esa posición. Por ejemplo, ponemos el cursor de la lista de *patterns* sobre la posición 01 que, en estos momentos, está ocupada por el *pattern* 00. Pulsamos ahora sobre la flecha vertical con el ratón, aumentando así el número de identificador de *pattern* a 01, de manera que si ahora reproducimos la canción con el botón *Play sng.* se reproducirá primero el *pattern* 00, que es el que creamos anteriormente, y después uno en blanco.

Seguramente nuestro segundo *pattern* será similar al primero en la base rítmica y cambiará la melodía, aunque no necesariamente. En caso de ser así sería interesante copiar parte del contenido del primer *pattern* en el segundo, así que veamos cómo se hace primero nos posicionamos en el primer *pattern*. Seleccionamos con el ratón (dejando pulsado el botón izquierdo) la porción que queremos copiar.

Pulsamos *Alt+F4* para copiarla en el buffer del sistema. Pasamos al segundo *pattern* y por último pulsamos *Alt+F5* ¡CON EL CURSOR EN LA POSICIÓN CORRECTA! para escribir el contenido del buffer en este pattern.

Ahora estamos listos para repetir el proceso de improvisación que antes explicamos con este nuevo *pattern* para completar el mismo. Una vez compuesta esta segunda parte de nuestra canción, podemos ya realizar una estructura más compleja, por ejemplo: 00 00; 00 00; 00 01; 00 01. Escuchamos el resultado mediante el botón *Play sng.*

Algunos consejos musicales

Aunque de una manera muy básica, ya somos capaces de crear canciones. La cuestión ahora es qué tipo de canciones queremos crear, ya que esto variará según el tipo de juego al que queramos ponerle una banda sonora.

Se suele asumir que los secuenciadores son sólo útiles para crear música Tecno, pero esto es completamente erróneo. Seleccionando los *samples* adecuados en cada caso y utilizando las técnicas necesarias, podremos crear desde potentes composiciones de Trash Metal hasta las más conmovedoras baladas Pop, pasando por estilos tan dispares como el Dance, el Rock'n'Roll o la música clásica.

Lo que es cierto es que a primera vista lo más sencillo parece optar por la música Tecno. Si tomamos la base rítmica que usamos anteriormente para practicar y ponemos la velocidad de lectura a BPM=125 y Spd=03 conseguiremos un típico ritmo de música Trance o House. Para completar el efecto, lo idóneo sería añadirle a la parte percusiva un *hithat* o *charles* a contratiempo, o bien a la misma velocidad que el bajo. Con contratiempo nos referimos, por ejemplo, a introducir toques de *hithat* en las líneas 4, 12, 20, 28, 36, 44, 52 y 60. Los contratiempos también se utilizan para hacer música Ska y Reggae, pero la diferencia es que para emular las baterías se introducen cajas, además de bombos, y normalmente las canciones se reproducen a velocidades menores, sobre todo en el Reggae.

Para la composición de música Rock en sus vertientes más potentes, necesitaremos *samples* de guitarras con distorsión y bajos contundentes, así como bombos agudos y platos de mayor envergadura como el crash o el ride. En estos casos no se hará casi nunca uso del contratiempo y, al igual que en todos los estilos citados hasta ahora, el compás utilizado será usualmente el 4x4, de manera que no deberemos variar el número de líneas de los *patterns*.

Si lo que queremos en cambio es componer músicas instrumentales como las que suelen acompañar a las grandes producciones cinematográficas, hay menos pautas a seguir, ya que los compases utilizados serán menos regulares. Los instrumentos más utilizados en estos casos son *strings*, órganos y sintetizadores, normalmente sampleados de instrumentos de síntesis de conocidas marcas. Es sobre todo en estas composiciones donde deberemos aprovechar todos los efectos que Fasttracker nos ofrece para conseguir un resultado más sorprendente.

Con lo que hemos visto en este artículo deberemos ser capaces de hacer nuestros primeros pinitos con Fasttracker. En los siguientes números trataremos a fondo todas las opciones del programa, pero para su correcta comprensión es necesario manejar con cierta soltura las funciones básicas.

Sergio Cánovas

esde
de Trash
novedoras
r estilos
ce, el
ásica.
e a pri-
parece
. Si
que usa-
practicar y
lectura a
eguire-
úsica
pletar el
dirle a la
charles
a misma
n contra-
ejemplo,
at en las
4, 52 y
mbién se
Ska y
es que
introdu-
bos, y
s se
menores,

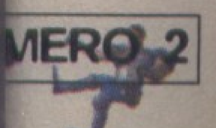
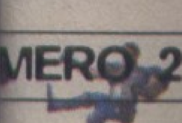
música

mples
y bajos
mbos
enverga-
e. En
nunca
gual
dos
zado
manera
núme-

ambio
mentales
añar a
emato-
seguir,
os serán
mentos
son
res,
instru-
das
as com-
s apro-

onse-
ndente.
en este
ces de
tos
entes
todas
pero
n es
soltu-

Cánovas



Prensa Técnica te facilita la llave para abrir la puerta al mundo de la informática a través de publicaciones especializadas y de propósito general.



Edita **PRENSA TÉCNICA**
Alfonso Gómez 42, Nave 1-1-2.
28037 Madrid
Tf: 91 3.04.06.22
Fax: 91 3.04.17.97

- **Si tu profesión o hobby es la informática, en Prensa Técnica tenemos el medio que estás buscando.**
- **Anímate, ya somos más de 250.000 lectores y seguimos creciendo.**



LA REVISTA QUE TE DA MÁS
MÁS PC, la revista informática para todos los públicos, con toda la información y actualidad en hardware, software, Internet, diseño, Linux, programación, videojuegos, multimedia, etc.

Pc
Incluye CD-Rom y libro técnico



CREAR ESTÁ EN TUS MANOS
3D WORLD está especializada en infografía y en general las 3D. Con la última actualidad en diseño gráfico, reportajes, técnicas, trucos y tutoriales de los programas de diseño y 3D más utilizados en el sector profesional.

Pc • Mac
Incluye CD-Rom

TU GUÍA PARA LA RED
INTERNET ONLINE se introduce en los recovecos de la gran Red mostrándole información rigurosa sobre aspectos técnicos, análisis de webs y herramientas. Incluye CD-Rom con navegadores, utilidades de correo, chat, etc.

Pc • Mac
Incluye CD-Rom

JUGANDO DURO
GAME OVER analiza los juegos de ordenador desde el punto de vista de los propios creadores. Toda la información técnica además de un análisis riguroso de las últimas novedades del mercado.

Pc
Incluye CD-Rom

LA NUEVA ERA DE LA FOTOGRAFÍA Y EL ARTE
FOTO ACTUAL Y ARTE DIGITAL, revista para profesionales y aficionados al diseño, maquetación y retoque fotográfico. La mejor forma de conocer toda la teoría y la práctica sobre las técnicas más utilizadas del momento.

Pc y Mac
Incluye CD-Rom

HAZ TUS PROPIOS VIDEOJUEGOS
DIV MANIA es la primera revista dedicada a aprender a programar videojuegos, abarcando todos los aspectos del desarrollo. Incluye CD-Rom con tres juegos programados por los lectores y demos de juegos profesionales.

Pc
Incluye CD-Rom



ELECTRÓNICA PRÁCTICA actual
INDICADOR BAROMETRICO
10 MONTAJES + CD-ROM GRATIS

LINUX ACTUAL
VIDEO FOR LINUX: THE PENGUIN CHANNEL

sólo LINUX
Cómo instalar Linux en tu ordenador en 15 minutos
Todo lo que hay que saber sobre Linux

LO ÚLTIMO EN TECNOLOGÍA
WINDOWS NT ACTUAL está destinada a profesionales del mundo NT. El modo más fácil para estar al día y conocer el entorno NT así como sus aplicaciones.

Pc
Incluye CD-Rom

POR Y PARA PROGRAMADORES
PROGRAMACIÓN ACTUAL te pone al día del mundo del desarrollo gracias a sus secciones principales dedicadas a la programación gráfica, Internet y sus lenguajes, desarrollo empresarial y nuevas tecnologías.

Pc
Incluye CD-Rom

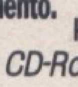
LA MÁS VENDIDA DE EUROPA
ELECTRÓNICA PRÁCTICA ACTUAL es la edición en castellano de la revista de electrónica más vendida de Europa. Contenidos prácticos de electrónica e informática con noticias, Internet y los montajes más ingeniosos.

Pc
Incluye CD-Rom

LO MEJOR, AHORA EN CASTELLANO
LINUX ACTUAL es la primera revista en castellano dedicada al GNU/LINUX: el sistema operativo de moda. Incluye artículos dedicados a todas sus áreas y un CD-Rom con las mejores distribuciones y novedades del momento.

Pc
Incluye CD-Rom

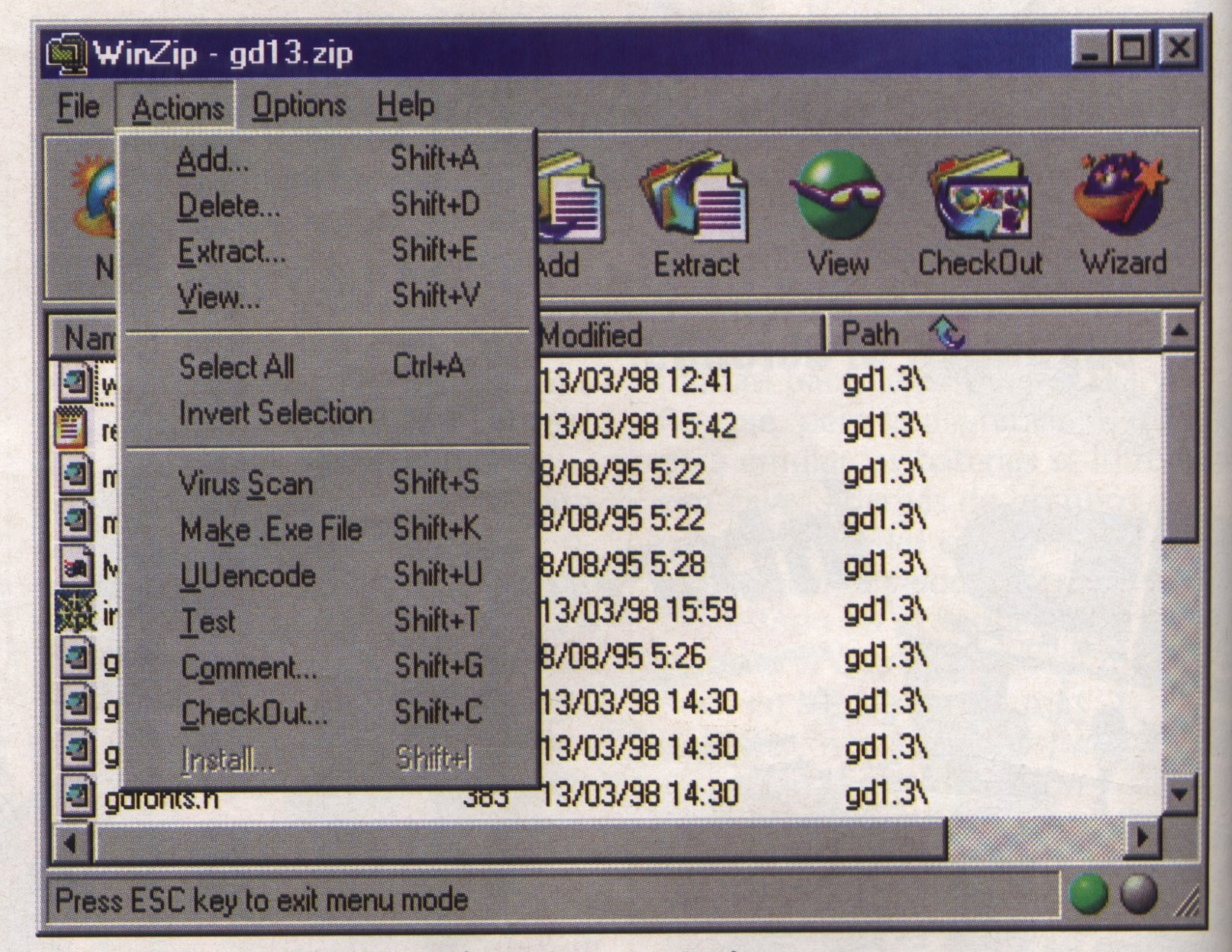
PENSADA PARA PRINCIPIANTES
SOLO LINUX es la mejor revista en castellano para el usuario principiante en el mundo GNU/Linux. En ella encuentra toda la información en forma de artículos de nivel básico. Incluye un CD-Rom con la distribución más fácil de instalar del momento.

Pc
Incluye CD-Rom


WinZip 7.0

La compresión de datos más fácil

Como pudistéis ver en la anterior entrega de esta revista, en la cual se comentó el uso del programa GetRight, esta sección de utilidades está dedicada a dar breves repasos a programas de fácil acceso (ya sea por Internet o por algún CD-ROM de programas Shareware). En esta edición se va a hablar de WinZip 7.0, en su versión de 32 bits (Windows 95/98/NT), aunque también existe una versión de 16 bits para Windows 3.x.



Aspecto que tiene el menú emergente Actions.

WinZip utiliza todas las posibilidades del entorno Windows para el manejo de archivos comprimidos ZIP y los formatos de compresión más comunes que se puedan encontrar en cualquier lado. Soporta los formatos TAR, gzip, UUEncode, XXencode, BinHex, y MIME. También soporta los ficheros ARJ, LZH, y ARC a través de programas externos.

Como programa suplementario está WinZip Internet Browser Support Add-On, programa que, junto con tu browser (Internet Explorer o Netscape Navigator) baja los ficheros ZIP de Internet abriéndolos una vez bajados, siendo esto mucho más cómodo para el navegante.

WinZip es capaz de crear paquetes comprimidos autoejecutables de gran utilidad para quien use habitualmente la herramienta

Otro programa suplementario en fase beta es WinZip Command Line Support Add-On, que te permite utilizar todas las funciones de WinZip en modo de ejecución automático (Ficheros *batch*, macros u otras situaciones en las cuales no se requiera el interfaz gráfico de WinZip, o se quieran realizar tareas de forma automática).


Instalación y configuración

La instalación del programa es muy sencilla, éste viene en un paquete de distribución autoextraible. Una vez instalado vemos que existen dos modos de funcionamiento del programa: el modo "Wizard" y el modo "Classic", que se pueden elegir indistintamente en cualquier momento.

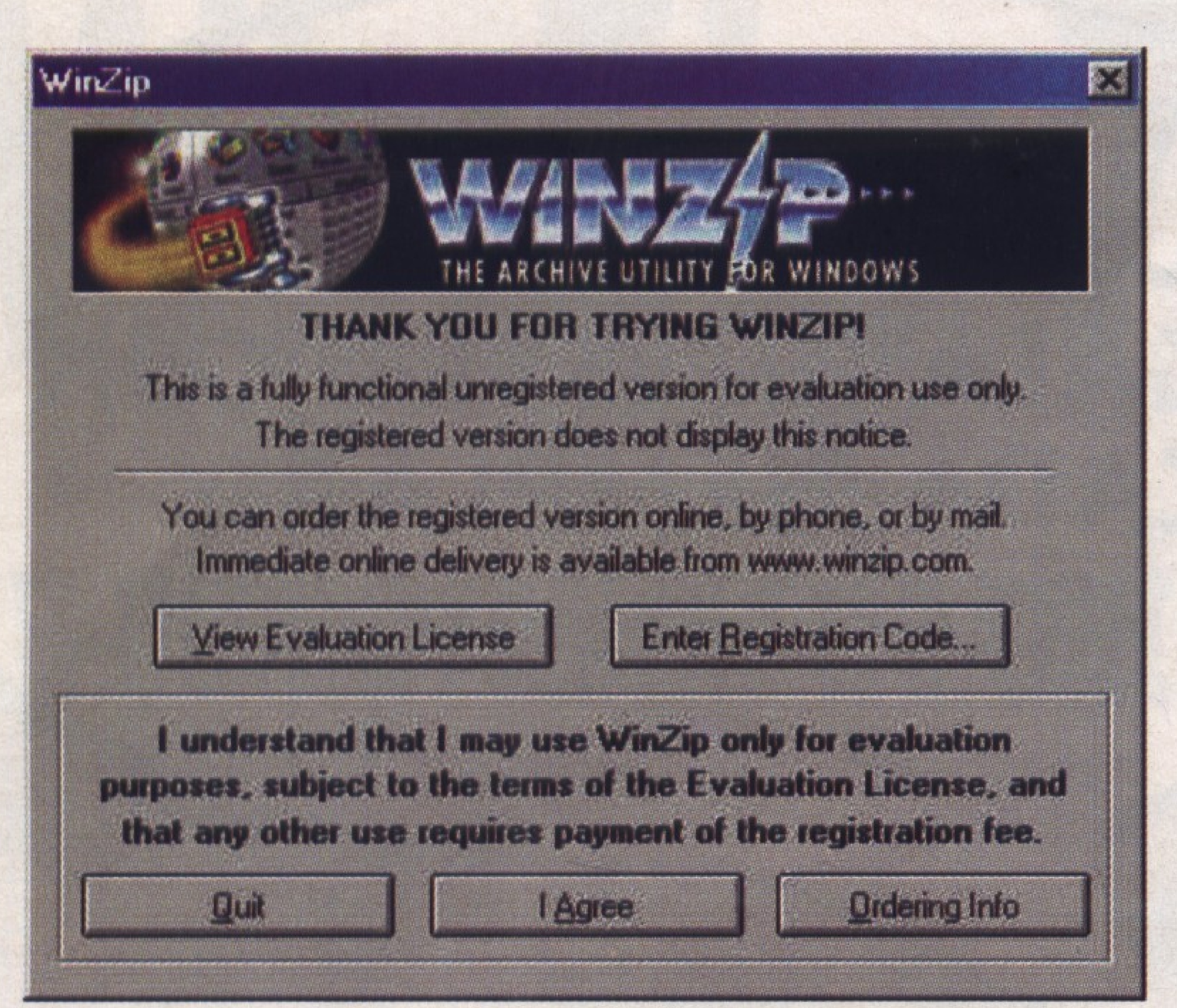
La configuración del programa es bastante sencilla, ya que una vez instalado, queda automáticamente configurado con las opciones más comunes. Únicamente resaltar la utilidad de incluir los programas externos ARJ, LHA y ARC en el menú *Program Locations*, ya que así posibilitamos a WinZip el manejo de los ficheros generalmente asociados a estos programas.

Un breve repaso

- *Modo "Wizard" de WinZip:* Este modo es ideal para usuarios que no estén muy familiarizados con el método clásico de WinZip. Es, no obstante, una manera de utilizar WinZip que no aporta todas las funciones que este programa es capaz de ofrecer y, sin embargo, muy fácil de utilizar. Está especialmente pensada para los usuarios inexpertos y para que éstos se vayan familiarizando con el sistema de WinZip.
Es un sistema que te va diciendo paso a paso lo que debes hacer. En un primer momento te pide que indiques el fichero que deseas procesar (si anteriormente no se ha utilizado el método Drag-and-Drop(Véase DivManía Número 1 Año 1), o se ha hecho un doble Clic en el archivo a descomprimir), una vez seleccionado se le indica la carpeta de destino y el programa hace el resto. Este método sólo soporta la instalación de software distribuido en archivos autoejecutables y la descompresión de archivos comprimidos.
- *Favorite Zip Folders:* no es más que una base de datos donde



CONSIGUE TU COLECCIÓN DE LIBROS TÉCNICOS DE PROGRAMACIÓN Y DISEÑO CON más PC
ME HE ENTERADO QUE AHORA MÁS PC, REGALA UN LIBRO CON CADA EJEMPLAR Y ¡AL PRECIO DE SIEMPRE!
más PC
a revista que te da MÁS: evista + libro + CD Rom
Calendario de entregas
Más PC nº2 Cómo programar en Java
Más PC nº3 Manual técnico Photoshop 4
Más PC nº4 Cómo programar en C
Más PC nº5 Manual técnico Autocad 14
Cómo programar en JAVA
Manual técnico PHOTOSHOP 4
Cómo programar en C
Manual técnico AUTOCAD 14
=+PC!
REPORTAJES
WebTV: la fusión de Internet y la TV en el hogar
Safedisc: la herramienta contra los "piratas"
MÁS CONTENIDO
Más información
Más formación
Más entretenimiento
INTERNET
IV Congreso Nacional de Usuarios de Internet
IRC en español
Microsoft FrontPage 2000
Webstorming
DISEÑO
Autocad 2000, el diseño gráfico del próximo milenio
World Builder
Kai's Power Tools 5
Adobe Premiere 5
Rhino 3D
MULTIMEDIA
Tutoriales Windows 98
Lorca en CD-Rom
Multimedia peques: cómo convertirse en estrellas del rock o aprender geografía
Cursos para ser un gran conductor o un pintor de renombre
COMUNICACIÓN
Comparativa de teléfonos móviles: los últimos portátiles del mercado
Exclusiva Más PC: Llega a España Web TV, el resultado de la fusión de Internet con la televisión
más PC
La revista práctica que te da más
Año 1 · Nº 3 · 995 pts.
HARDWARE Todo en impresoras, grabadoras y ordenadores de bolsillo
SOFTWARE La última versión de Lotus Notes/Domino/Designer
INTERNET IV Congreso Nacional de Usuarios de Internet
DISEÑO AutoCAD 2000, el diseño gráfico del próximo milenio
VIDEOJUEGOS Rollcage, las dos caras de la velocidad
MULTIMEDIA Tutoriales Windows 98 Lorca en CD-Rom
LINUX Todos los programas de diseño gráfico para Linux
PROGRAMACIÓN Java 2, el estándar definitivo para desarrollo web
CONCURSOS
Descubrimos todas las novedades
El mejor hardware del 99
Impresoras
Web TV
Móviles
Handhelds
CD-ROM
Curso de Excel paso a paso
Bryce 2
Demos: File Maker Pro 4.1
Isla Soft Contawin 2000
Antivirus Panda Platinum 6.0
Demos juegos: Rollcage, MIG-29 Fulcrum y Mask of Eternity
HARDWARE
El mejor hardware 99
Grabadoras
Impresoras
Ordenadores de bolsillo
SOFTWARE
Lotus Notes/Domino/Designer versión 5
MediaScan
Teamroom 1.5
Prens@ Técnic@
Edita PRENSA TÉCNICA
Alfonso Gómez, 42. Nave 1-1-2.
28037 Madrid
Tf: 91 3.04.06.22
Fax: 91 3.04.17.97
No te pierdas el número 4 de Más PC
A la venta el 1 de Abril sólo por 995 ptas.



Es posible realizar el registro vía Internet.

se integran todas las carpetas de las cuales alguna vez hemos descomprimido algún fichero, haciéndonos la vida más fácil, si, por ejemplo, solemos bajar todos los ficheros de Internet a una misma carpeta (ej. Downloads).

- *Ficheros autoejecutables:* WinZip Self-Extractor Personal Edition es un pequeño programa que viene con el paquete de WinZip 7.0, que permite que los archivos comprimidos con WinZip no necesiten del propio programa para ser descomprimidos.

Si se utilizan discos removibles para la realizar el proceso de compresión, WinZip lo manejará todo simplificando las tareas que tiene que realizar el usuario

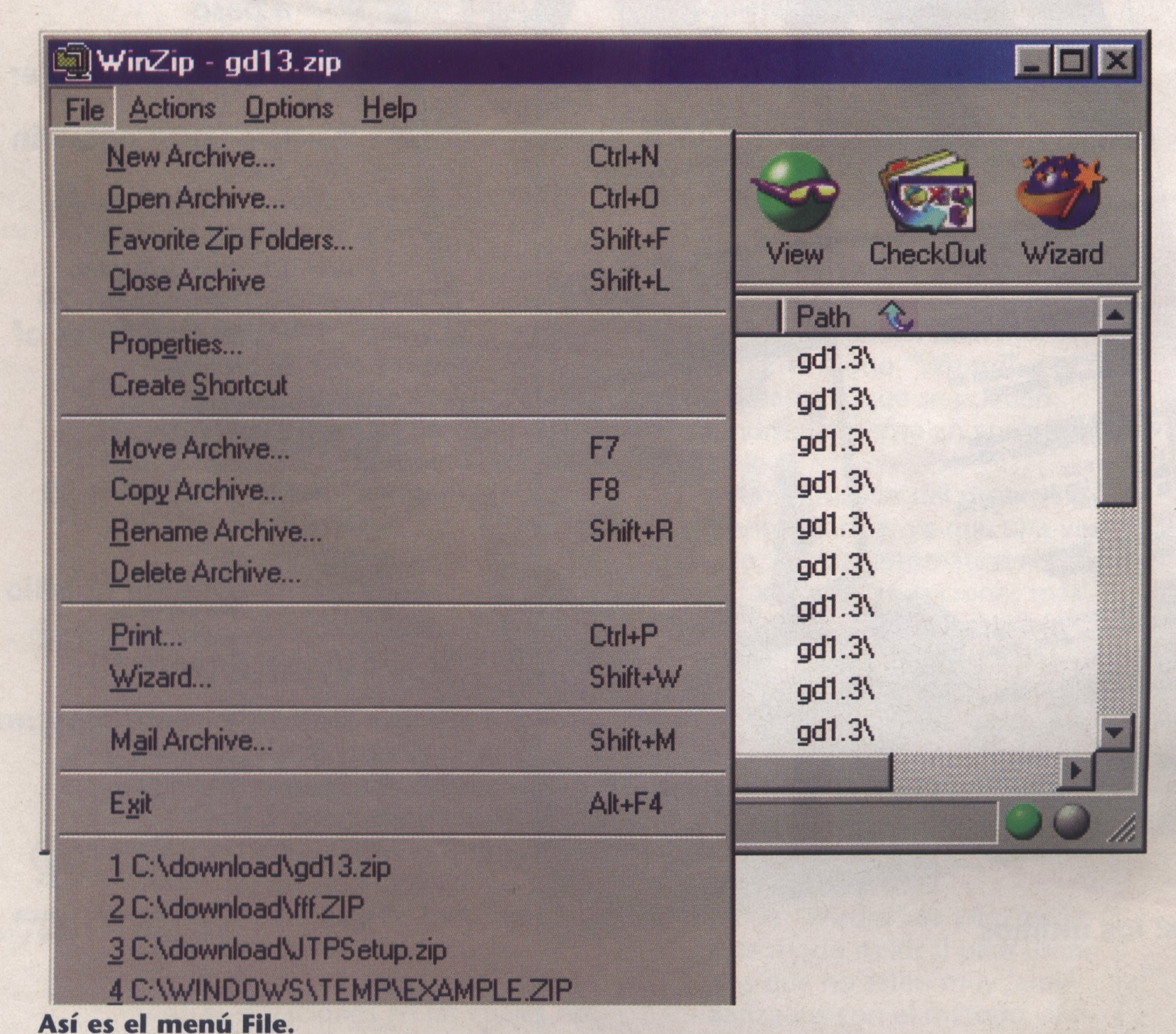
Este sistema es ideal para mandar archivos comprimidos a personas que no posean el programa para descomprimirlos.

- *Soporte antivirus:* WinZip puede ser configurado para que al abrir, o bajarte de Internet un archivo, ejecute automáticamente un antivirus sobre el archivo en cuestión, esta opción soporta los antivirus más comunes del mercado.
- *Modo "Classic" de WinZip:* en el modo Clásico de WinZip nos encontramos con el típico interfaz al cual nos tenía acostumbrados Nico Mak Computing, Inc. en sus anteriores versiones de WinZip, pero con la diferencia de unos nuevos iconos indicativos, éstos son: *New, Open, Favorites, Add, Extract, View, CheckOut y Wizzard*, y unos menús desplegables en la parte superior de la ventana que son: *File, Actions, Options y Help*. Como es común en casi cualquier programa de Windows, los iconos de la parte media no son mas que ShortCuts o atajos a algunas de las funciones más usuales del programa, siendo todas ellas accesibles a través de los menús desplegables de la parte superior. Por esta razón en este artículo únicamente describiremos las funciones de los menús desplegables.
- *Menú File:* este menú está dedicado al manejo de ficheros, se puede abrir un fichero o crear uno nuevo, cerrarlo o cambiarlo de nombre, moverlo, copiarlo o incluso mandarlo vía E-mail.
- *Menú Actions:* añadir(*Add*), borrar(*Delete*), extraer(*Extract*) ficheros o ver(*View*) los contenidos de un fichero comprimido son las principales acciones, aunque también se puede comprobar la integridad de un fichero comprimido(*Test*), hacerlo ejecutable (*Make .EXE*) o pasarlo al formato ASCII UUEncode.
- *Menú Options:* aquí es donde vamos a poder personalizar el programa a nuestro gusto, la configuración (Vista con anterioridad), las contraseñas (*Passwords*) de los ficheros creados (Opcional), el orden en el que queremos que aparezcan los ficheros en la ventana de WinZip, etcétera.
- *Menú Help:* que todavía tenemos dudas, o que queremos saber la dirección de la página web de WinZip, no tenemos más que acudir a este menú, que posee una ayuda bastante detallada.
- *La ventana de WinZip:* seguramente estéis acostumbrados a la utilización del Explorador de Windows, y a su manejo *drag-and-drop*. Pues bien, la ventana de WinZip se comporta de manera idéntica al Explorador, podemos seleccionar un archivo de esta ventana y hacer con él lo que más nos guste, renombrarlo, moverlo o borrarlo, es tan sencillo como hacer un clic.

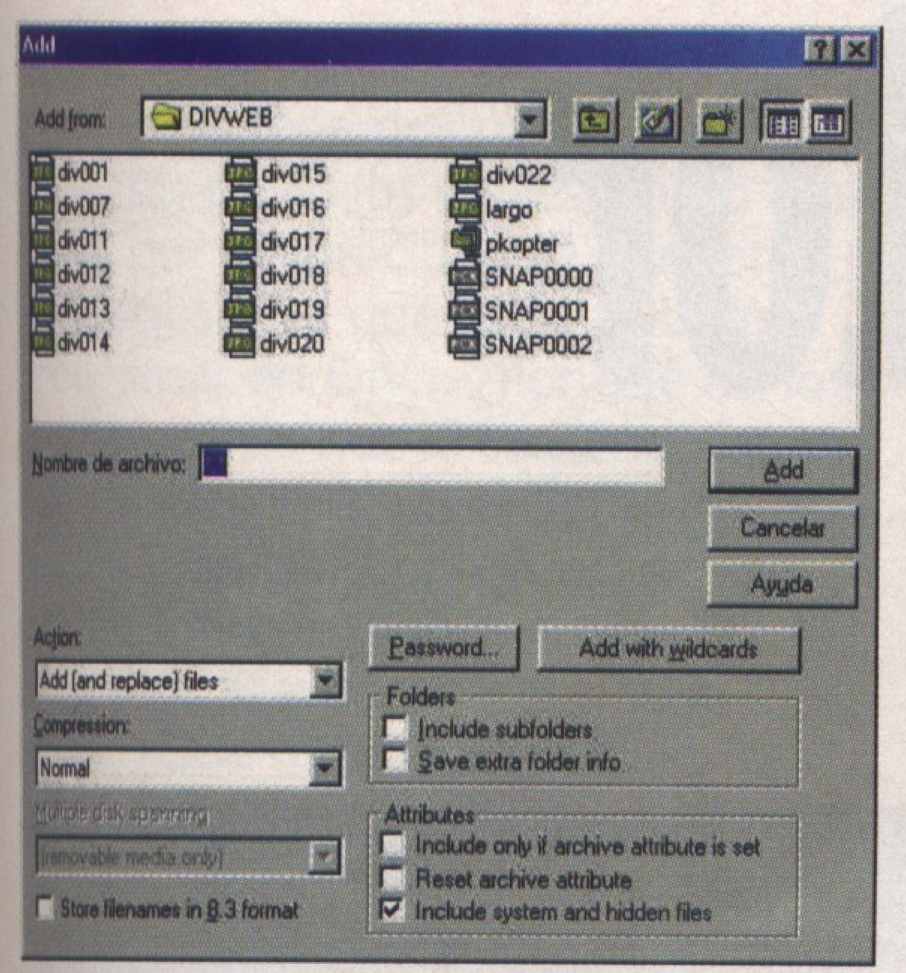
Creación de nuevos archivos comprimidos

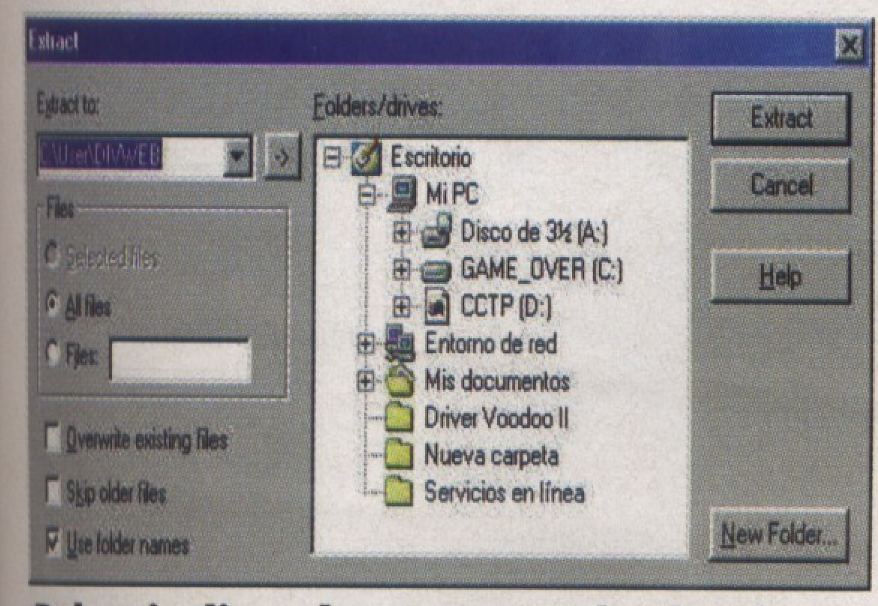
Para crear un nuevo archivo ZIP hay que seleccionar *New Archive* del menú desplegable *File*. Esto activará otra ventana, donde deberemos introducir el nombre del fichero a crear, además de la carpeta donde el fichero será creado. Una vez hecho esto, presionamos *OK* y nos aparecerá otra ventana donde tendremos que indicar los archivos que queremos comprimir, así como el método de compresión y las distintas opciones:

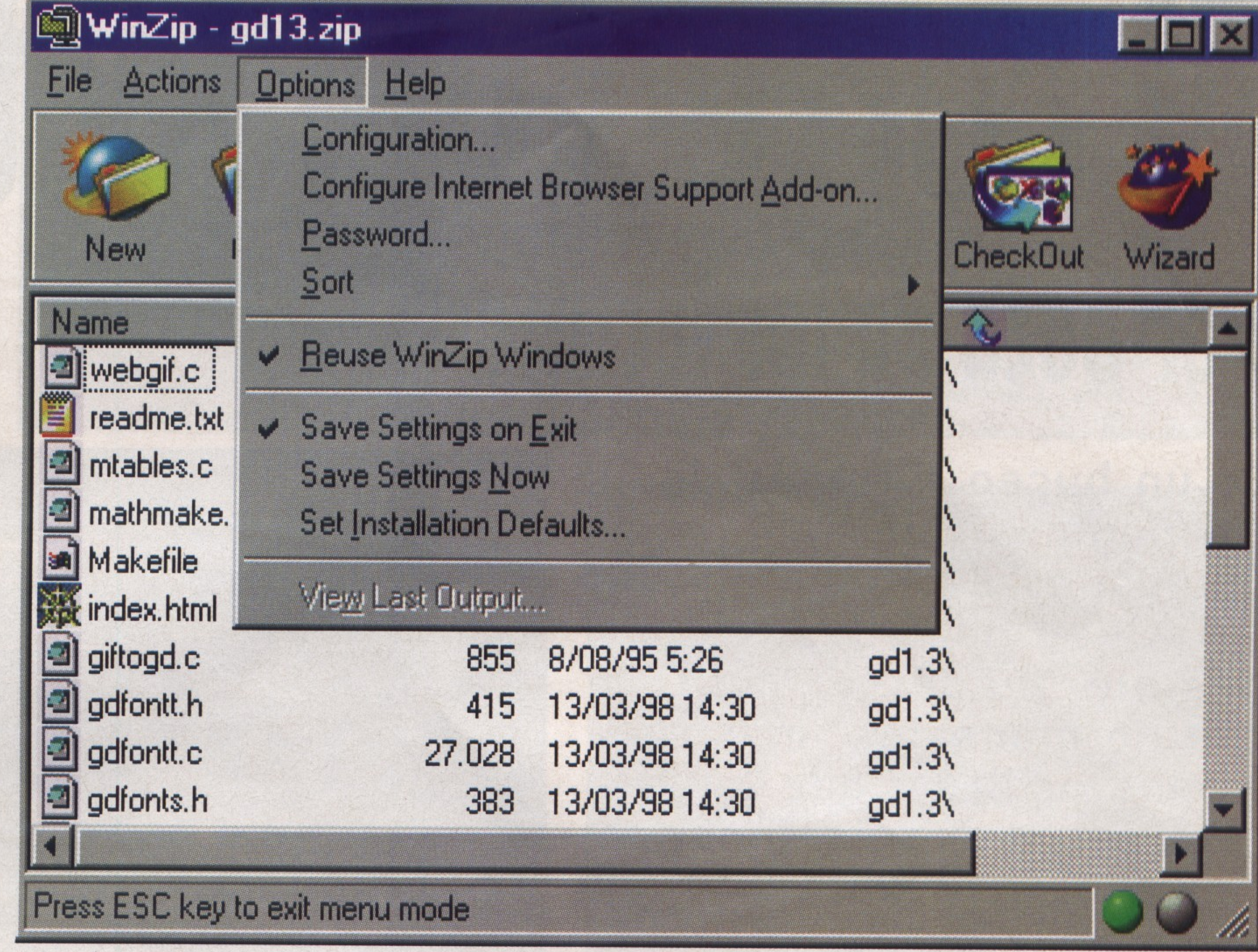
- *Recurse Subfolders*: esta opción, si la marcamos, incorporará al archivo comprimido todas las subcarpetas, incluidos los archivos, de las carpe



Así es el menú File.


Realizar los archivos .zip es sencillo.


Debes indicar la carpeta de destino.



Menú Options.

tas que seleccionemos para comprimir.

- *Save extra folder info*: si activamos esta opción, se guardará en el archivo comprimido el *path* absoluto de cada archivo que comprimamos.
- *Store filenames in 8.3 format*: opción que *trunca* los nombres largos de archivo, exclusivos de los Windows de 32 bits al formato de 8 caracteres de nombre y 3 de extensión antes de guardarlos en el archivo comprimido, haciendo así posible su lectura en un sistema operativo de 16 bits.
- *Include only if archive attribute is set*: únicamente añade al fichero comprimido los ficheros que tengan el indicador de lectura/escritura o modificado (+A). Para más información, *Propiedades del menú Archivo del Explorador de Windows*.
- *Reset archive attribute*: quita el indicador de modificado a los ficheros que han sido añadidos al archivo comprimido.
- *Include system and hidden files*: añade también los ficheros ocultos y de sistema.
- *Password*: por si se quiere añadir una contraseña al fichero comprimido.

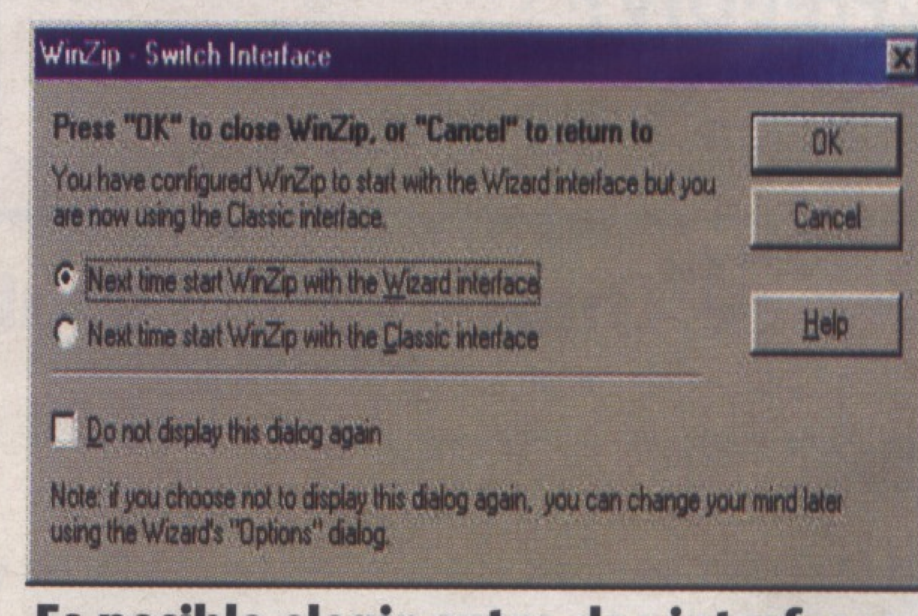
Una vez elegidos los ficheros y carpetas que queremos añadir al nuevo fichero, y puestas a gusto todas nuestras opciones, es el momento de terminar presionando el botón *Add*, que creará el fichero y volveremos a la ventana original, pero esta vez con el fichero que acabamos de crear, abierto.

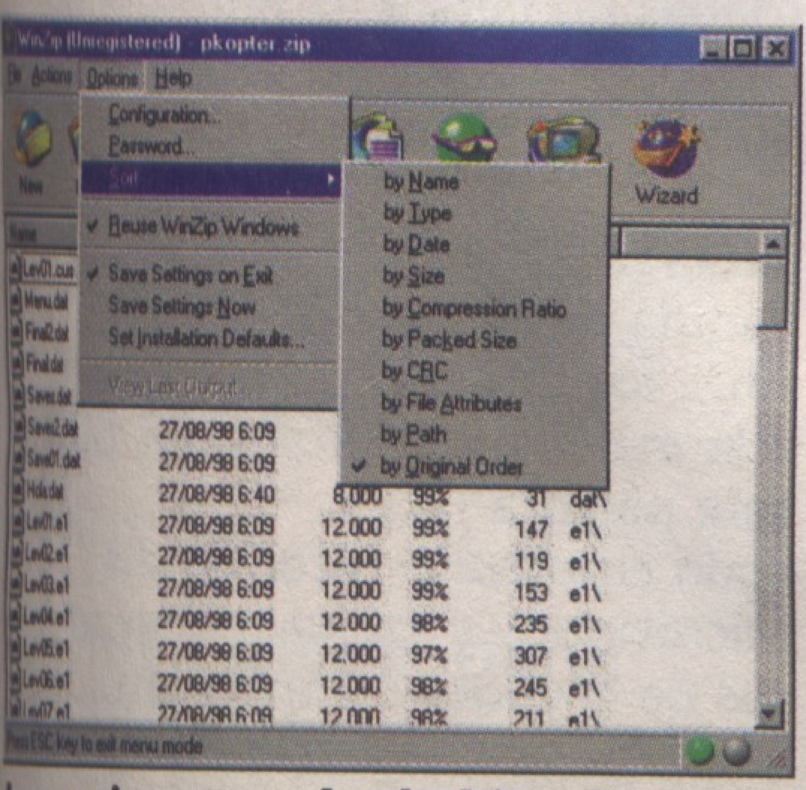
En el caso de que queramos abrir un fichero comprimido no tenemos más que seleccionar la opción *Open Archive* del menú *File* y nos encontraremos en la misma situación que al crear un archivo: nos aparecerán los contenidos de éste en la ventana de WinZip.

Extracción de ficheros comprimidos

La extracción de ficheros es muy sencilla, al instalarse, WinZip asocia automáticamente los tipos de ficheros ZIP con el programa, por lo que haciendo simplemente un doble clic sobre el fichero comprimido, se ejecuta automáticamente el programa. Nos vamos al menú *Actions* y seleccionamos la opción *Extract*. Nos aparecerá una nueva ventana donde deberemos indicarle la carpeta de destino, las distintas opciones y ficheros a extraer (que anteriormente habremos seleccionado con el método habitual de Windows en la ventana de WinZip). Presionamos el botón *Extract* y se extraerán automáticamente.

Javier Fernández


Es posible elegir entre dos interfaces.


Las opciones son las habituales.

La interfaz ideal es la clásica.

Información

Si queréis más información sobre WinZip, los datos de *Nico Mak Computing* son:

E-Mail: *support@winzip.com*
Web: *http://www.winzip.com*
Correo: *PO Box 540*
Mansfield, CT 06268 (USA)

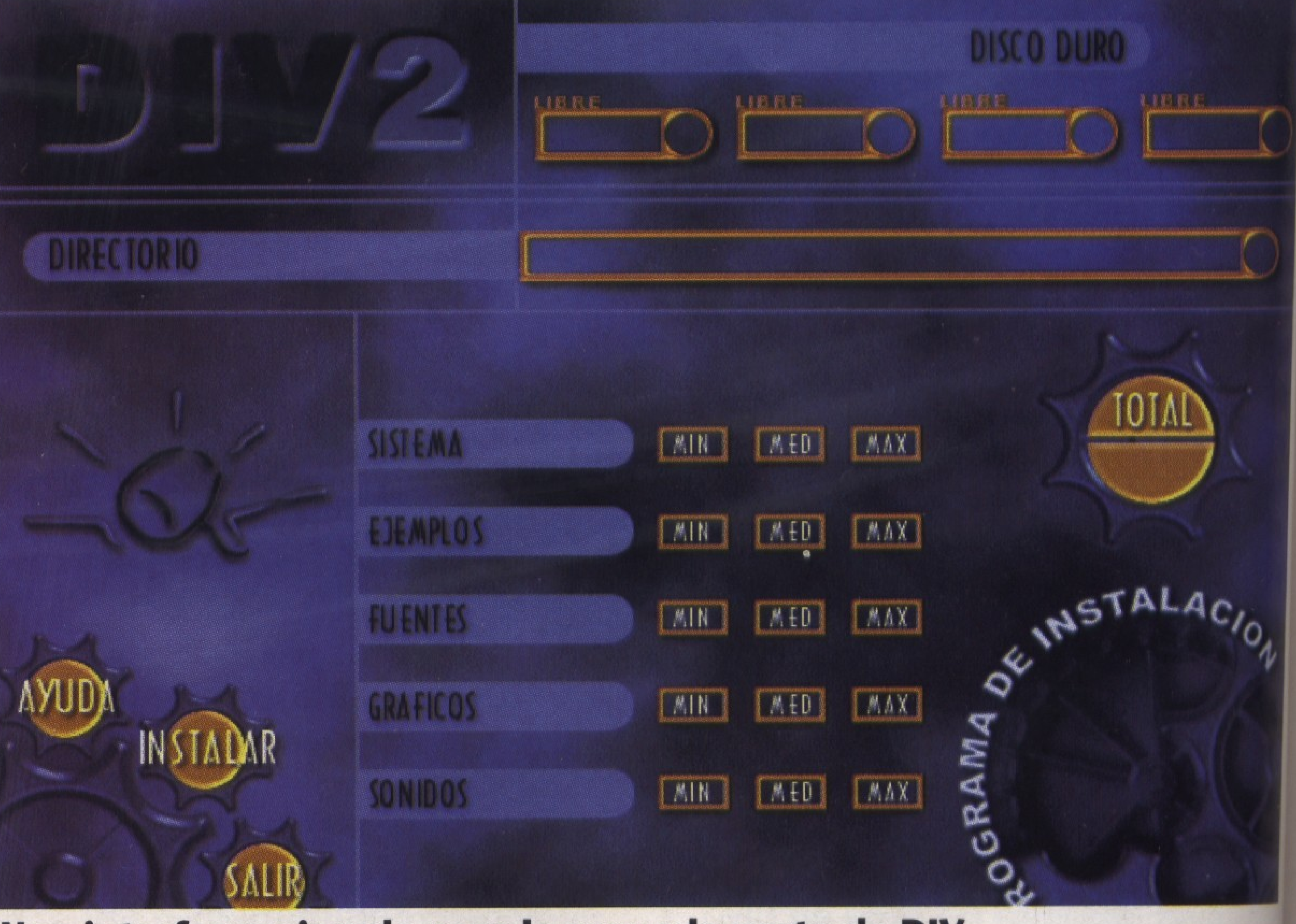
Hasta el próximo numero de la revista.




El correo del lector

Por boca de los lectores

En un buceo a través de las páginas que están dedicadas a DIV dentro de la Red de redes, es posible encontrar multitud de mensajes, más o menos ocultos, que anuncian eventos relacionados con nuestro entorno de programación favorito. Esta sección la completaremos, de todas formas, con los mensajes que nos lleguen a la redacción.



Una interfaz mejorada para la segunda parte de DIV.

Parece que uno de los comentarios que más se ha hecho versa acerca de la incierta salida de DIV 2, así como del segundo número que tenéis entre las manos. Nosotros ya estamos aquí, y DIV 2, si a estas alturas no está ya en la calle, poco le falta, desde luego. Pero echemos un vistazo a lo que se cuece entre los usuarios de DIV.

¿FreeDIV?

Página de XIMO, martes 30 de marzo de 1999.

Bueno, vamos a hablar de FreeDiv: la nueva revolución en el canal #div. La idea de FreeDiv nació el domingo pasado (28), según creo. Scooby lo contó a los que estaban en el canal y el resto ya es historia...
En cosa de dos días han creado una *mailing list* en freediv@onelist.com (ve a la página de DIV, sección listas de correo si quieres suscribirte) y la web oficial, aunque creo que es provisional.
Sí, pero ¿qué es FreeDiv? No, aún no lo he pillado; en el canal hablaban el domingo de un independizarse de DIV y en la web he leído algo de hacer DIV para Windows. Sea como sea, esto es una pequeña lista de las cosas que quieren hacer:

- Permitir formatos gráficos comprimidos, como GIF y JPG (a ver cómo lo hacemos).
- Ampliar las funciones de DIV Games Studio.
- Permitir el uso de funciones de otros lenguajes de programación, como C o Pascal, por medio de alias.
- Hacer que todo eso funcione bajo Windows de una manera fácil y cómoda.

Y toda esta movida se ha montado porque ya hay muchas personas que están hartas de la política que siguen los de Hammer Technologies. Si la idea prospera (ganas no faltan), la potencia de DIV podría verse aumentada al 400%, por lo menos.

Nostalgia

Página de Ferminho.
Ah... los usuarios de DIV suscritos a la *mailing list* conocemos bien esta palabra ;) ...
Todo empieza al recordar una MicroHobby y los juegos tan adictivos de la época de Spectrum. Nuestros ojos se llenan de lágrimas. ¡Aaahhh...! Y dado que todos

Sesiones de charla en el canal

Nick del moderador	Temas	Horarios	Periodicidad
KILLERS	Iniciación / Scroll	18:00-19:00	Semanal

Comentario: Creo que es importante dedicar un mínimo de tiempo a una especie de curso de iniciación.

Nick del moderador	Temas	Horarios	Periodicidad
TIZO	Modos 7	Viernes Noche	Semanal

Comentario: Espero que la gente se anime...

Nick del moderador	Temas	Horarios	Periodicidad
DIVmaster	Inteligencia Artificial	Viernes de 12 a 14h	Semanal

Comentario: Para los que sepan y para los que no, al final llegaremos a programar a Hal 9000 ;D.

Nick del moderador	Temas	Horarios	Periodicidad
Nauta	Inteligencia Artificial	De 19 a 21h	Diaria

Comentario: Sesión diaria de Inteligencia Artificial.

echamos tanto de menos esos juegos... y no tenemos ganas de conectar todos los cables, ya no funciona por desgracia, o se perdieron los juegos... ya que todos sabemos programar en DIV... ¿Por qué no los recreamos de nuevo haciendo nuestras propias versiones? Sería una buena idea crear un grupo de programadores de juegos antiguos, llevar varios proyectos entre nosotros... ¿Qué os parece? Incluso podría llevarlo en esta *page*... si os parece bien... @

Night Cowboys

Bueno, los Unif Studios están desarrollando ahora, conjuntamente con otros proyectos más ambiciosos, una especie de arcade de plataformas al estilo del clásico *Sunset Riders*. También es de una especie de pistoleros, pero con muchas novedades, por supuesto. Incluye un método de programación similar al del juego *Abuse*, en el que el protagonista consta de dos procesos: uno, las piernas, y otro, la parte de arriba. Esto evita tener que hacer muchos gráficos con todas las posiciones de apuntar a todos los lados, recargar, etc., mientras se anda, se salta, etc.

Lista de E-mails comunes

Ésta es la forma mas rápida y cómoda de pedir sugerencias, preguntas, soluciones a tus problemas con DIV y su programación. Si te suscribes formarás parte de una lista de correo común, lo que quiere decir que si mandas un mensaje con alguna duda, la podrán visualizar muchísimos DIV adictos que intentarán solucionártela. Y si te gusta solucionar las dudas de los demás, también podrás hacerlo. Suscríbete a la lista de E-mails comunes en la cual recibirás y podrás enviar E-mails a *oneline*. Una vez suscrito podrás enviar los E-mails a *canaldiv@onelist.com* y entre todos intentaremos solucionar tu problema, duda, sugerencia, etc. ¡Gracias por colaborar!

Se buscan programadores

Busco programadores de DIV para proyecto interesante en Barcelona y provincia. También busco gente de 20 a 28 años para trabajar a tiempo parcial que tenga buenos conocimientos en lenguajes de programación e INET.

xare@drac.com

Se busca gente para aprender

Se busca gente para aprender en esto del DIV. No hace falta tener especiales cualidades, simplemente tener ganas e ilusión. Más información en *www.fortunecity.com/skyscraper/whitecat/516*.

primis_soft@latinmail.com

Grupo de programación sobre DIV

Estamos formando un grupo de programación sobre DIV. El grupo se llama DIVer_soft. Necesitamos Grafistas 2D y 3D, programadores que controlen IA, ya que el primer proyecto es un juego de estrategia. Todos los interesados que se pongan en contacto conmigo.

Jons@netspain.com

Se busca gente de alrededores de Bilbao

Se busca gente de los alrededores de Bilbao para desarrollar algún proyecto con DIV. Sólo se necesita muchas ganas e ilusión.

molpe@clientes.euskaltel.es

DIVeros en Bilbao y Logroño

Me gustaría saber de DIVeros que fuesen de Bilbao o de Logroño para intercambiar todo aquello referente a DIV que se nos ocurra.

5jolopez@rigel.deusto.es

Proyectos en Visual Basic, Delphi, Turbo Basic y DIV

Somos un pequeño grupo de jóvenes programadores. Para los interesados en realizar proyectos en Visual Basic, Delphi, Turbo Basic o DIV, escribir. Próximamente página web oficial.

alfaro@accesosis.es

Se buscan programadores en DIV y/o Visual Basic

Soy un aficionado a DIV que busca gente que sepa programar en DIV o en Visual Basic 5 para formar 2 grupos, uno para juegos y otro para programas de Windows. No nos vendría mal algún grafista, preferiría a gente de Oviedo o alrededores y menores de 18 años, preferiblemente gente de mi edad, o sea, entre 14 y 16 años.

kai@lettera.net

Aventura Gráfica

Busco colaboradores para realizar una aventura gráfica con DIV. Proyecto serio.

fnp@jet.es

Juego de Rol - "TORAM"

Se buscan grafistas y programadores para realizar un juego de rol basado en el argumento de NightWolf. Interesados mandar un E-mail con título TORAM.

txa66666@teleline.es

Redacción DIV Manía

Esperamos vuestros comentarios

La dirección de correo para enviar vuestras ofertas, demandas, preguntas o respuestas es la siguiente:

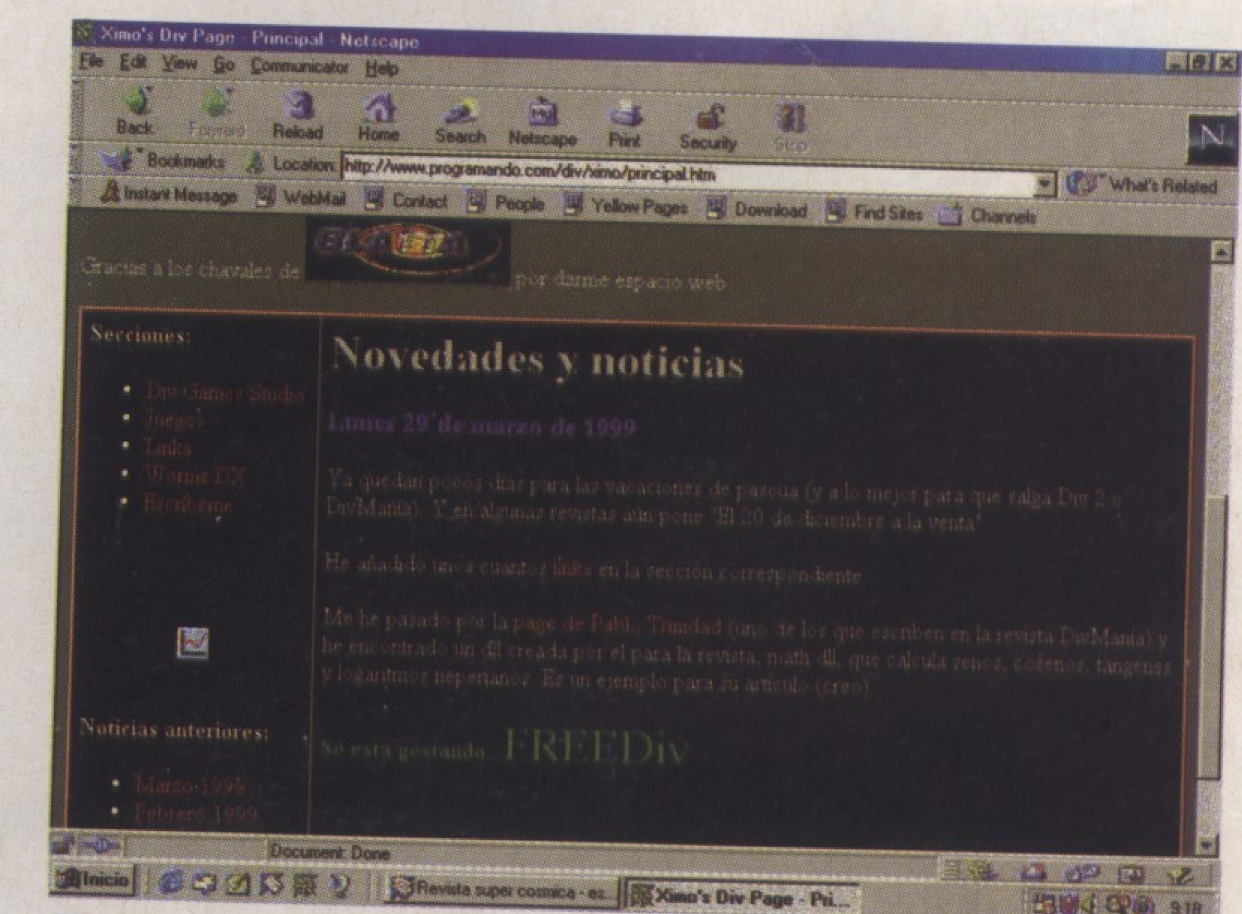
Revista DIV Manía
Sección Correo
C/ Alfonso Gómez, 42, nave 1-1-2
28037 Madrid

También podeis usar la siguiente dirección de correo electrónico, indicando en el "subject", como tema, "DIVManía – Sección: Correo". Todos los "chateros" ya tendréis noticias nuestras a estas alturas.

divmania@prensatecnica.com



Un entorno de trabajo muy especial, en una esperada segunda edición.



Dentro de esta página se encuentran algunas referencias al concepto de FreeDIV.

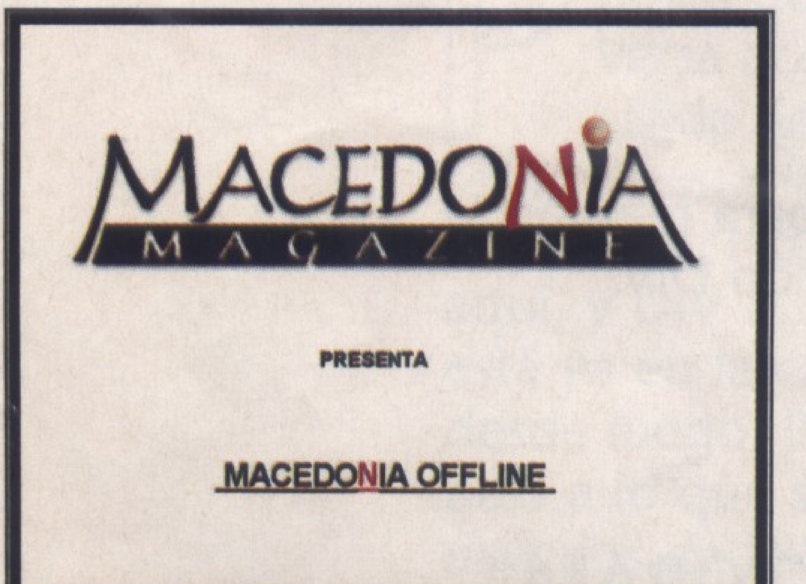


Contenido CD-Rom



Se ha hecho imprescindible la edición de un CD-Rom conjuntamente con las revistas del mundo informático, aún más en ésta, dedicada a la programación de juegos. Sin este soporte de almacenamiento de información no podríamos facilitaros las utilidades, librerías y el software en general para que os apliquéis en nuestros cursos. Ahora podréis instalar en vuestros equipos las versiones más actuales de las aplicaciones dedicadas a programación y aprender de los juegos que se han alzado con los premios de este número.





DEMO DIV GAMES STUDIO 2

El primer número de nuestra revista venía acompañado de un CD-Rom en el que incluíamos una demo de Div Games Studio. Ahora os proporcionamos el esperado Div Games Studio 2, la nueva creación de Hammer Technologies para desarrollar vuestras ideas hasta convertirlas en juegos.

La nueva versión de este entorno de desarrollo para juegos incluye nuevas opciones, utilidades y atajos para hacer de la programación algo accesible a los no iniciados, incluyendo una ayuda interactiva al servicio del usuario. Para usarla, únicamente tenemos que presionar la tecla F1 cuando el cursor del editor esté encima de la palabra reservada sobre la que tengamos alguna duda. Para los que ya tienen sobre sus espaldas horas y horas de trabajo programando, esta nueva herramienta será de gran utilidad.

La versión de Div Games Studio 2 que proporcionamos viene como demostración, por lo que no podrás acceder a todas sus posibilidades. A pesar de ello, es eje-







cutable y podrás utilizarla y programar en ella sin ningún problema. Si quieres disfrutar de Div Games Studio 2 en su totalidad tendrás que comprar el producto.

MACEDONIA MAGAZINE

Ya puedes disfrutar con el número siete de Macedonia Magazine, una publicación digital en español que nos ha llamado la atención por su sencillez y versatilidad. En ella encontrarás gran cantidad de información, además de links a otras páginas de Internet.

Los contenidos son tan variados que te darán qué leer durante largo rato. Las secciones *Recuerdos de 8 bits* y *Zona de juegos* están dedicadas al mundo del entretenimiento informático. También podrás acceder desde aquí a clases de los diferentes lenguajes de programación.

SHAREWARE

BROWSERS:

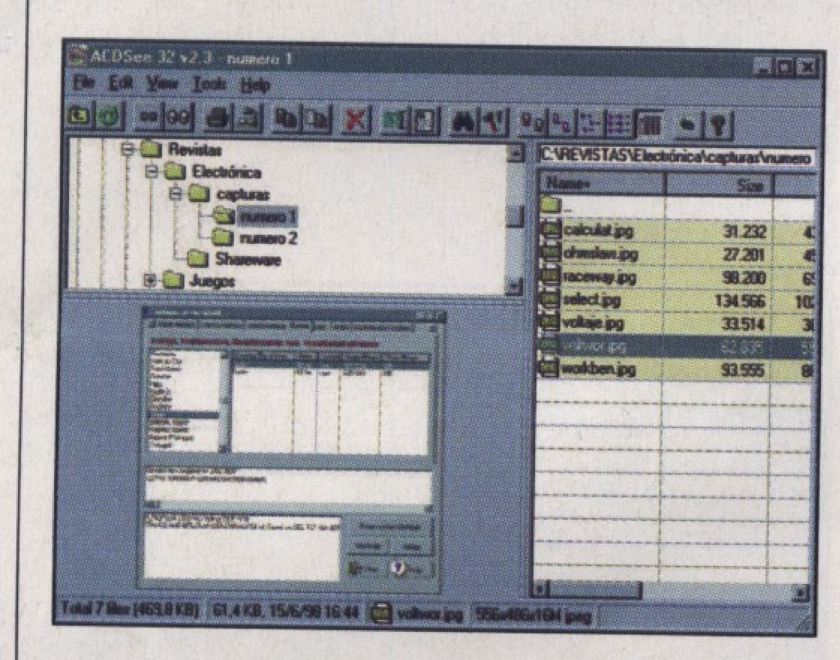
NETSCAPE COMMUNICATOR 4.5, INTERNET EXPLORER 4.01

DRIVERS:

DIRECTX 6.0

ACDSEE 2.3.

Visor gráfico de gran rapidez que permite mostrar en pantalla una *preview* de los ficheros de un directorio o, si lo prefieres, mostrarlos uno tras otro automáticamente. El programa soporta los siguientes formatos gráficos: BMP, GIF, JPG, PCX, PNG, TGA, TIF y Photo-CD. También incluye opciones zoom, impresión y conversión de la imagen actual en fondo de pantalla de Windows.



ANTIVIRUS PANDA PLATINUM

Util antivirus que detecta los virus que conoce y los nuevos. Puede trabajar residente y verificar todos los archivos que se baje de Internet.

ACROBAT READER 3.0.

Lector de ficheros con formato PDF.

GRAPHICS WORKSHOP.

Visor gráfico que incluye una gran cantidad y diversidad de opciones. Entre ellas están: ProcesosBatch, GIF animados, PSD de Photoshop, CAM de Casio, KDC (Kodak Digital Camera), FIF (Fractal Image), opciones de escáner y muchas otras opciones.

PAINT SHOP PRO 5.01.

Programa shareware de retoque fotográfico por excelencia. Incluye la mayoría de los filtros con los que cuentan los programas comerciales. En esta nueva versión puede utilizar los Plug-ins de Photoshop.

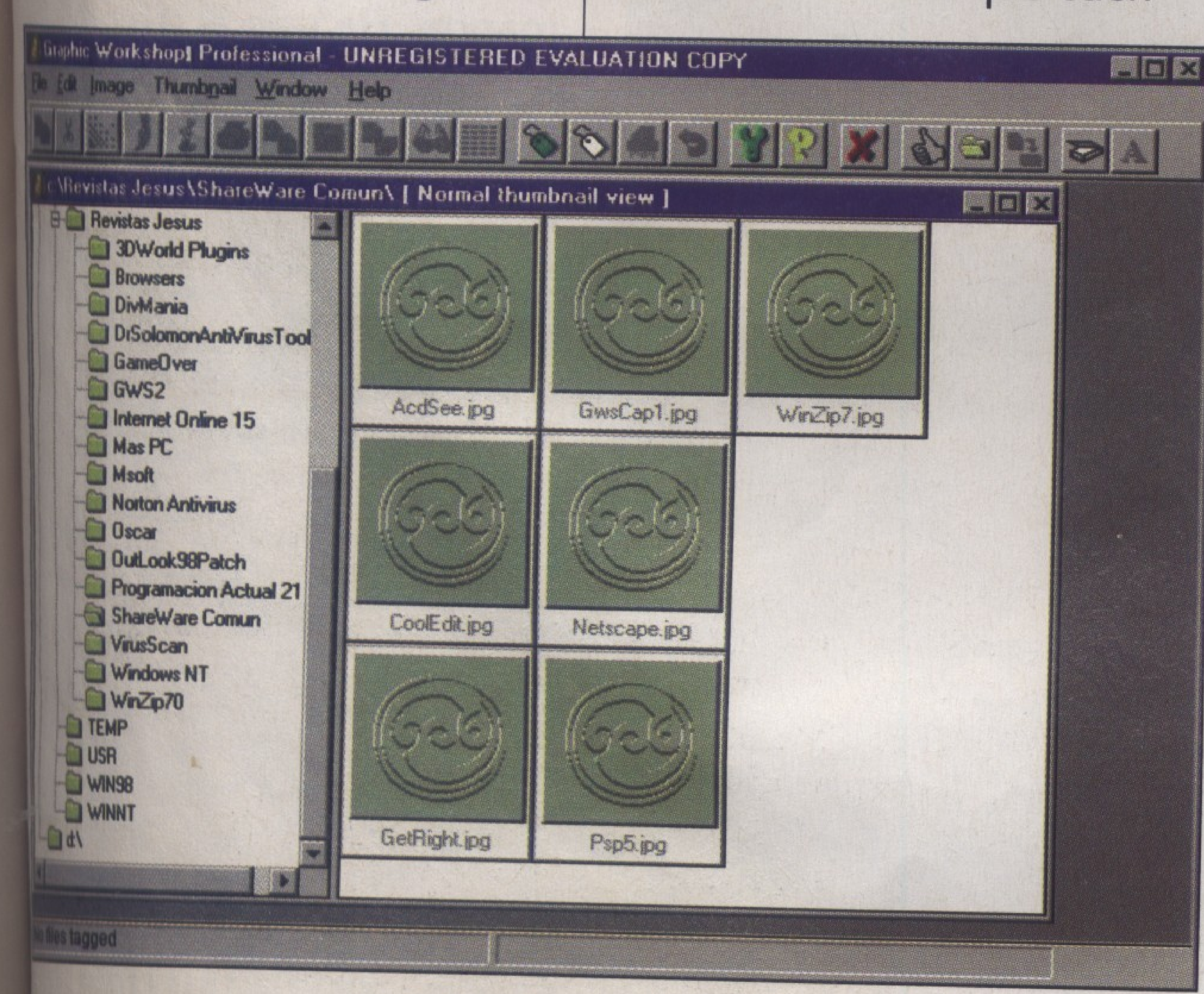


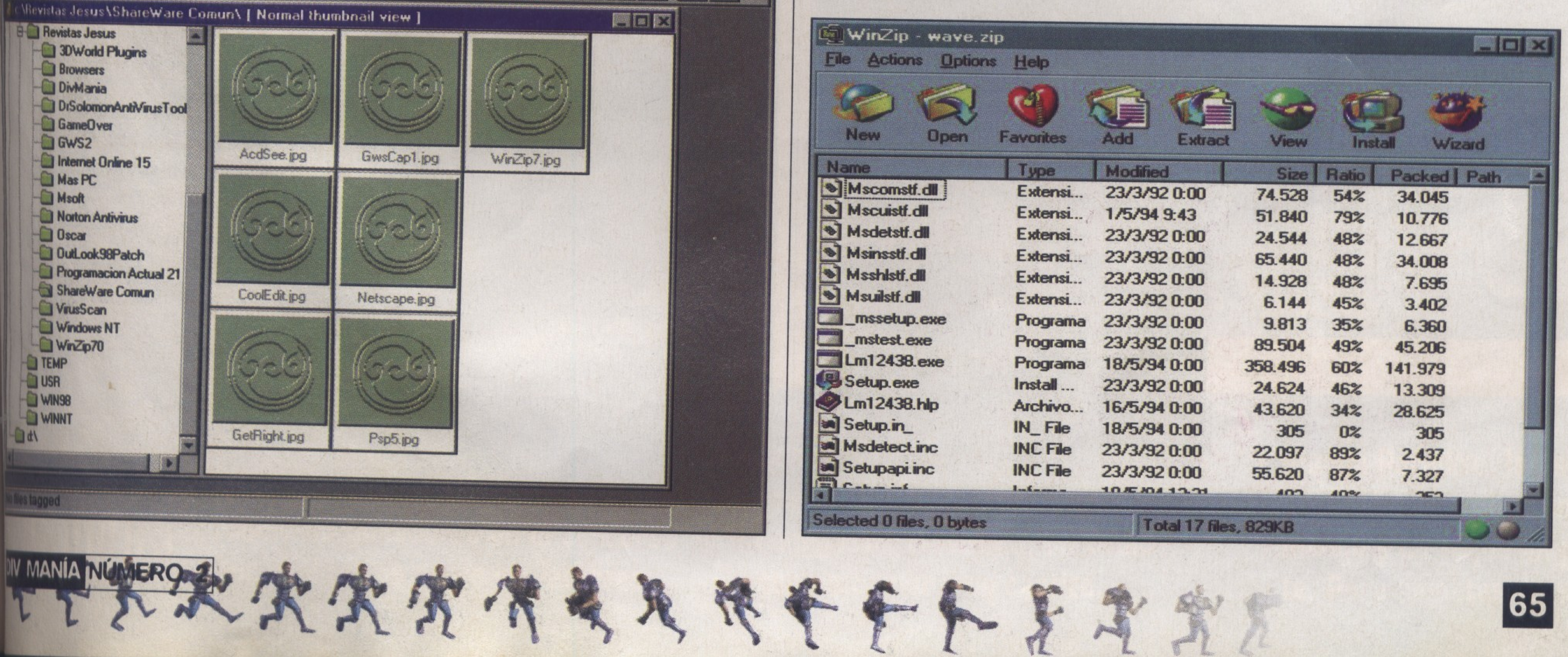
SEA GRAPHICS VIEWER 1.3.

Uno de los mejores visores gráficos para MS-DOS. Permite previsualizar de las imágenes y admite resoluciones de hasta 800x600 en millones de colores y 1280x1024 en 256 colores. Soporta los formatos gráficos JPG, TGA, BMP & RLE, TIFF, PCX, GIF, LBM (IFF), PSD, CEL, Dp2E-LBM, BBM & PCC, PNM (PBM, PGM, PPM) y PNG, y de animación FLI,FLC y AVI.

VIRUSSCAN DE MCAFEE 4.02

Es uno de los antivirus más extendidos. Es capaz de detectar y limpiar una gran cantidad de virus, también es capaz de trabajar residente en memoria y detectar cualquier virus que entre en el sistema por cualquier medio, ya sea por disco, Internet o red local, el VShield se activa y avisa que hay un virus.



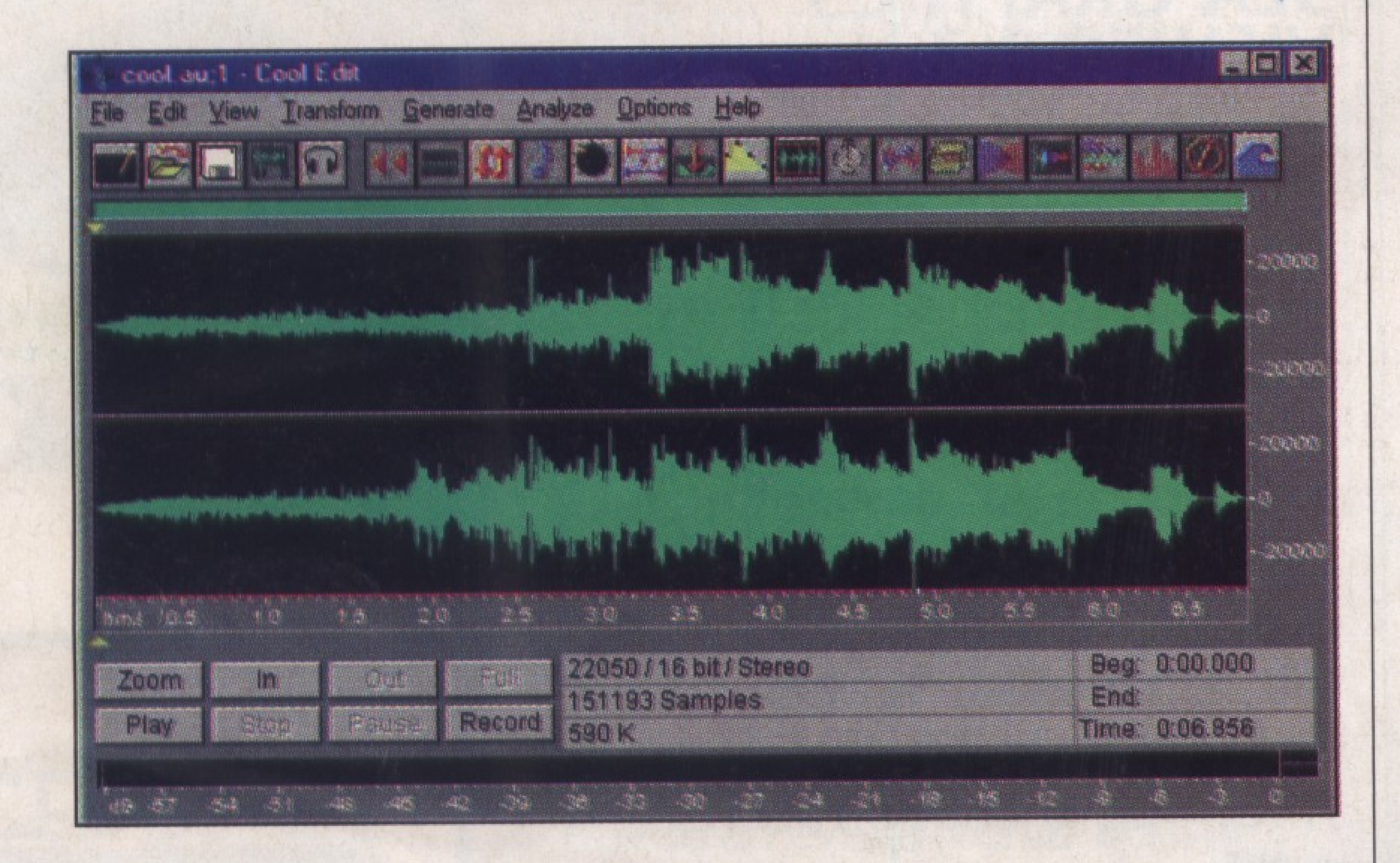


WINZIP 7

La última versión del compresor de archivos más conocido. Permite trabajar con diferentes formatos como ARJ, LZH, ARC, TAR, TGZ, Unix Compress, UUEncode, XXencode, BinHex, MIME, y como no, ZIP.

COOLEDIT 96

Este programa de edición de audio permite utilizar las funciones más cotidianas como cortar, pegar, grabar o reproducir todo ello. Además permite seleccionar bloques del fichero de sonido. Proporciona opciones para modificar frecuencias o crear efectos.



LIBRERÍAS

Hemos incluido librerías útiles para quienes deseen internarse en el apasionante mundo de la programación de videojegos. Poco a poco iremos ampliando las librerías del número anterior de Div manía, en las que pudiste encontrar todo tipo de fuentes, sonidos y texturas. Curiosea un poco y verás el contenido de las nuevas librerías que te proporcionamos en este CD-Rom.







JUEGOS

Dentro de este directorio encontrarás los juegos premiados en este número de la revista. A ver si el próximo mes sois vosotros quienes tenéis un juego publicado en esta sección. Veamos los ganadores de este número.



COMANDO PELOTA

Sencillo pero entretenido. En *Comando Pelota* debes proteger a tus pelotitas en medio de un bombardeo. Espero que os guste, ya que a nosotros nos ha gustado muchísimo; basta con ver que se ha llevado el primer premio.

FUMIGASIÓN GALÁSTICA

Catalogar este juego es muy sencillo, marcianitos con humor. Ha sido el segundo juego seleccionado. Ahora podéis acceder a su interior sin tener que copiar líneas de comandos.

MANGA PARADISE

En *Manga Paradise* debes descubrir la imagen oculta. ¿Será realmente un paraíso? Compruebalo tú mismo. Al acabar se nos premiará con la imagen de una bella señorita de los famosos cómics manga. Ha sido el tercero premiado.





NÚMERO 2

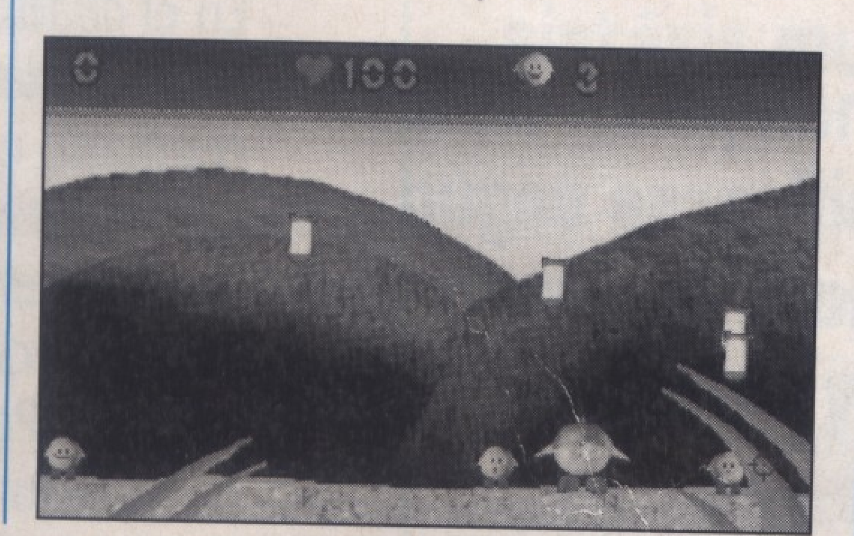
Sumario

Curso de programación y concurso

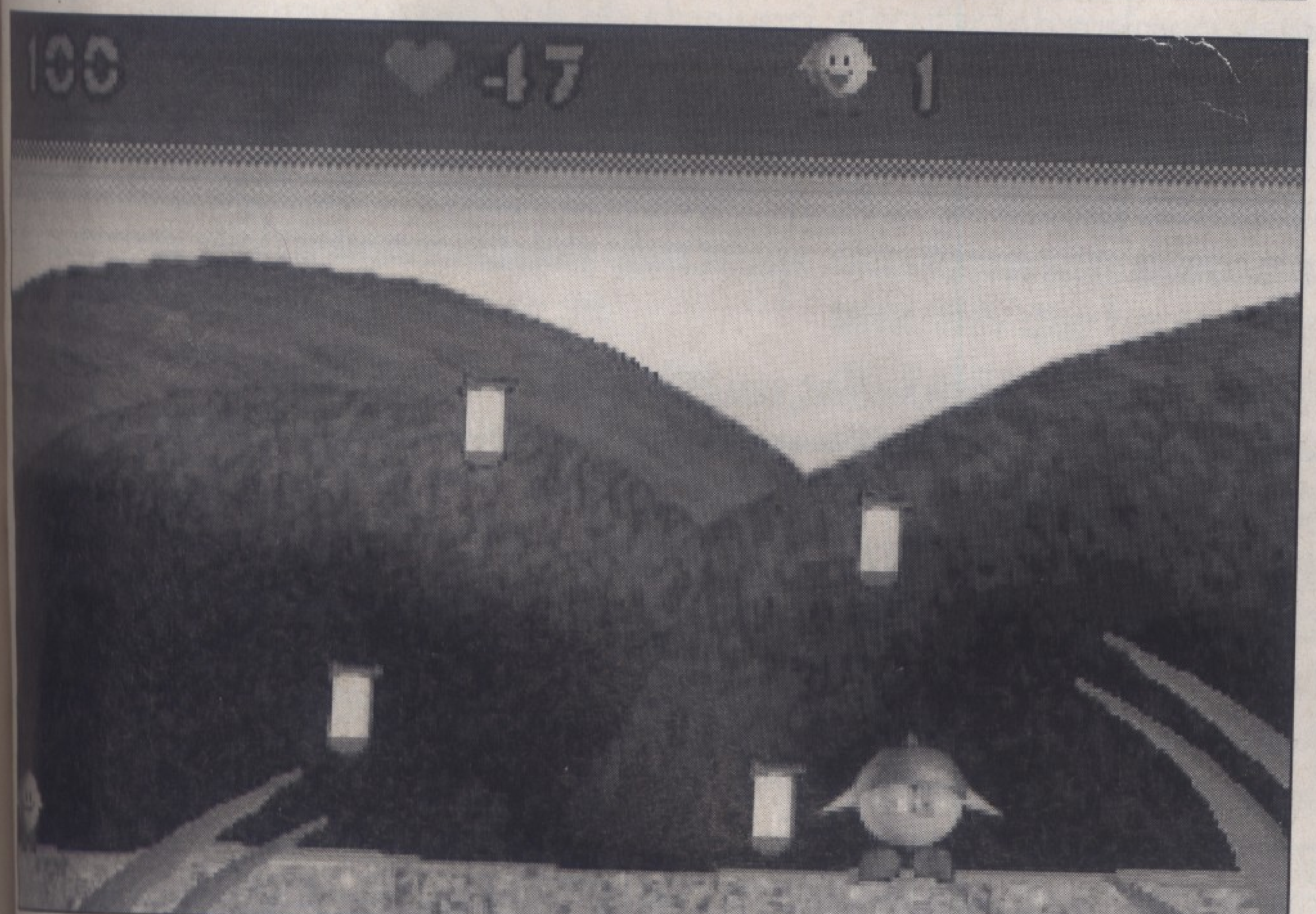
En este nuevo número de la revista Div Manía no puede faltar una de las secciones que ha obtenido una mayor aceptación entre todos nuestros lectores. Como ya sabéis, está íntegramente dedicada al desarrollo y a la programación de videojuegos, dentro de la cual podéis encontrar la continuación de los tres cursos que publicamos el número anterior de vuestra revista preferida.
El primero de ellos ha sido íntegramente dedicado a la programación básica (es decir, a la programación que está basada en los algoritmos) y se la ofrecemos a todos aquellos lectores que quieran empezar a dar sus primeros pasos en este campo o bien deseen refrescar los conocimientos que ya tienen dentro del mundillo de la programación general.

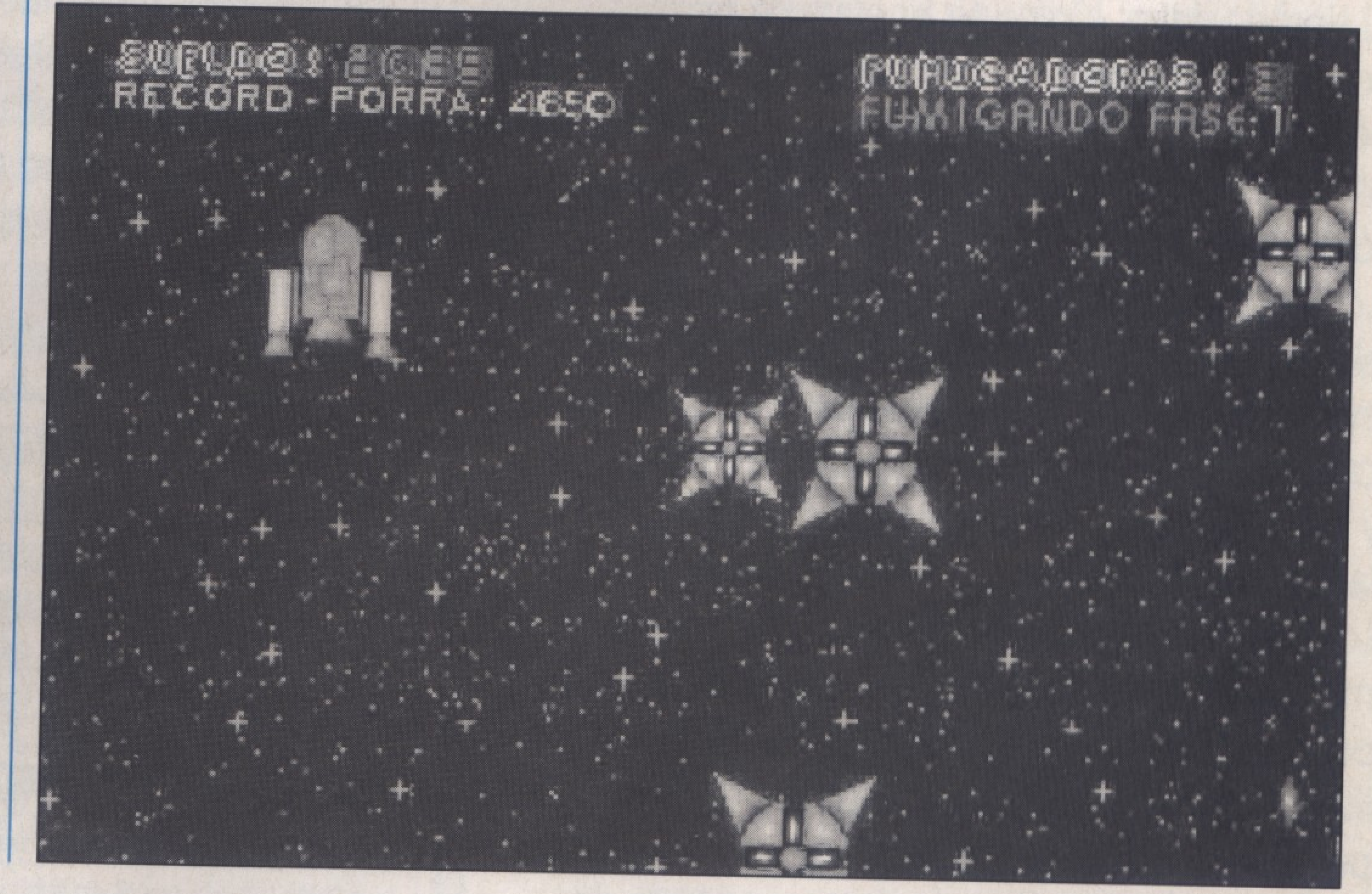
Así pues, el segundo curso que ponemos a vuestra disposición esta vez trata, nuevamente, de la programación en C. Son dos intensas páginas para un lenguaje de alto nivel que es utilizado por buena parte de los programadores actuales.
Y el tercero nos nos introduce de lleno en el mundo del Ensamblador. Como sabéis, se trata del lenguaje con el que funciona el propio microprocesador central del ordenador. DIV es una buena herramienta de trabajo, desde luego, pero será capaz de alcanzar una efectividad mucho mayor si se combina con los lenguajes de programación que os proponemos, complementos ideales para este ya de por sí excelente entorno de desarrollo de videojuegos.
Continuamos, por lo demás, con el **concurso de programación** en el que, como ya sabéis, premiamos a los juegos de mayor calidad entre todos los que llegan hasta nuestra redacción. En las páginas siguientes encontrarás los tres seleccionados para este número con los comentarios que han realizado los propios desarrolladores de los juegos.
El primer premio del concurso consta de **25.000 pesetas**, y el segundo y tercero de **20.000**. No está nada mal para estos pequeños juegos que realizamos todos con la ayuda de DIV.
Para poder participar no tenéis más que escribir al E-mail de la revista: divmania@prensatecnica.com, presentando vuestro juego, o bien enviarlo a la dirección que escribimos más abajo. Todos los programas serán bien recibidos.

C/ Alfonso Gómez, 42
Nave 1-1-2
28037 Madrid-España









stacamos

En nuestro CD de portada incluimos las siguientes demos, que han resultado ganadoras en el concurso que realizamos entre los lectores:

- Comando Pelota, un programa muy original y adictivo.
- Fumigasión Galástica, un título con un humor muy castizo.
- Manga Paradise, la demo de un programa que tiene un aire de manga y un toque un poco picante. Para amantes de cómic nipón.

Con ayuda de los algoritmos

Este artículo presenta las instrucciones de control de programa, completando la descripción de los elementos básicos que componen un lenguaje de programación.

Se pueden definir los tipos de datos elementales como aquellos que no necesitan de otros tipos para ser definidos, siendo la base para poder formar tipos compuestos de datos, que se obtendrán relacionando los tipos de datos elementales. Normalmente, aunque pueden variar según el lenguaje, los tipos de datos elementales son: numéricos, lógicos y carácter, estando divididos los datos de tipo numérico en enteros y reales.

El tipo de datos entero representa un subconjunto finito de los números enteros (no poseen parte decimal). El rango de valores de un dato entero estará entre los valores *-2n-1 y 2n-1-1*, donde n representa la longitud de la palabra del procesador. Fuera de este rango no es posible representar ningún otro número entero.

Los datos de tipo real representan un subconjunto, también finito, de los números reales, pudiendo disponer de decimales, permitiendo representar mayor cantidad de números que el tipo entero, pero a costa de perder precisión en el cálculo.

El tipo de datos lógico (o booleano) permite representar los valores verdadero y falso. Normalmente, este tipo de datos está asociado al valor de una condición o expresión de tipo lógica o relacional.

Los datos de tipo carácter permiten representar el juego de caracteres que utiliza el ordenador; evidentemente, entre ellos se encontrarán los caracteres alfabéticos, los caracteres numéricos y los símbolos que normalmente se utilizan en la vida diaria *(*, +, <, $ etc)*.

Los programas a través de sus instrucciones manipulan datos que pueden clasificarse según puedan o no modificarse en constantes (o literales) y variables. Las constantes que no pueden modificarse a lo largo de un programa pertenecerán a un tipo de dato: si se trata de un tipo numérico tomará la forma de un número (45, es una constante numérica de tipo entero); si es de tipo carácter ésta se encontrará encerrada entre delimitadores, normalmente apóstrofos, por ejemplo, *'a'* representa la letra a. Las constantes de tipo lógico solamente pueden ser verdadero o falso.

Una variable no es más que un lugar de la memoria central que permite almacenar valores del tipo de la variable; dicho contenido puede ser consultado o modificado por las instrucciones del lenguaje. En todos los lenguajes, y antes de que cualquier instrucción pueda hacer referencia a una variable, hay que declararla. La declaración de una variable consiste en asociarle un identificador válido de variable y un tipo de datos, lo que permitirá delimitar el tipo de valores que puede almacenar. La sintaxis del lenguaje establecerá los lugares permitidos para la declaración de variables.

Los programas manipulan datos que pueden clasificarse según puedan o no modificarse en constantes y variables

Un identificador de variable, o nombre de variable, es normalmente una cadena de caracteres numéricos, alfabéticos y el carácter subrayado (_), que debe comenzar por un carácter no numérico; esto permite diferenciar los identificadores de variable de los literales numéricos; además, los identificadores pueden distinguirse de los literales de tipo carácter debido a que estos últimos están delimitados. Es necesario recalcar que una constante "vale lo que vale", a diferencia de las variables que cada vez que son referenciadas valen su contenido. Puesto que el programador puede seleccionar los identificadores de variable es conveniente que se seleccione un nombre que esté acorde con el contenido o la función que realice.

Un tipo de datos, además de tener asociado un rango de valores admisibles, posee un conjunto de operaciones en las que constantes y variables de dicho tipo pueden aparecer como operando; es posible combinar operandos y operadores, bien sean constantes, bien sean variables, para construir expresiones. En estas expresiones podrán aparecer además paréntesis y funciones. El resultado de evaluar una expresión será un valor que dependerá tanto del tipo de los operandos como del tipo de las operaciones. En el cuadro 3 se muestran los operadores más comunes clasificados según el tipo de resultado que ofrezcan cuando se evalúan.

Con el fin de poder asignar valores a las variables en todo lenguaje existe la instrucción de asignación. Tomando como referencia el lenguaje Pascal, la instrucción de asignación es :=. Para su utilización se dispone en la parte izquierda de esta instrucción la variable a la que se asignará el valor resultante de evaluar la expresión que esté situada en la parte derecha de dicha instrucción. Cuando ésta termine de ejecutarse, el valor anterior de la variable se

LOS TIPOS DE DATOS ELEMENTALES SON NUMÉRICOS (ENTEROS,REALES), LÓGICOS Y CARACTERES.

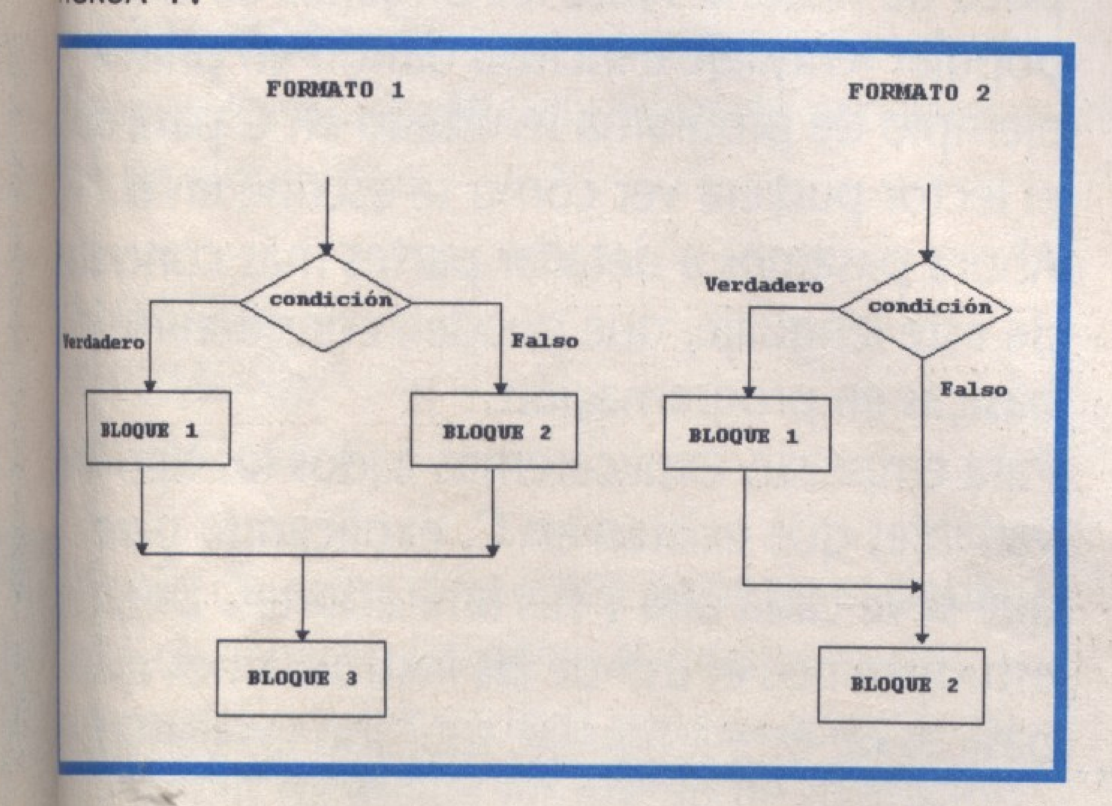
Operador	Tipo de operandos	Tipo de resultado	Significado
+	Enteros o reales	Entero o real	Suma
-	Enteros o reales	Entero o real	Resta o signo negativo
*	Enteros o reales	Entero o real	Producto
div	Enteros	Entero	División
/	Reales	Real	División
mod	Enteros	Entero	Módulo
<, <=	Enteros, reales, carácter	Lógico	Menor, menor que
>, >=	Enteros, reales, carácter	Lógico	Mayor, mayor que
=	Enteros, reales, carácter	Lógico	Igual que
not	Lógico	Lógico	Negación
or	Lógico	Lógico	Unión
and	Lógico	Lógico	Intersección

ierde, es decir, una variable no puede tener dos valores de forma simultánea. s posible asignar a una variable el valor esultante de una expresión en la que ntervenga ella misma; cuando esto ocurre, a la ora de evaluar la expresión se tomará el valor ctual de la variable pasando posteriormente a omar el valor resultante de la evaluación de la xpresión. Así, si la variable *numero* tiene valor 5 antes de ejecutar la instrucción *numero:= numero+3*, tras su ejecución la variable tendrá el valor 8.

En todo lenguaje existe la instrucción de asignación para identificar los valores de los variables

Todos los lenguajes de programación, sin excepción, contemplan una sintaxis muy férrea. Este hecho se refleja en que cualquiera de ellos presenta palabras reservadas, cuyo uso está predefinido por el lenguaje, identificadores asociados a elementos que puede definir el programador a su conveniencia, constantes, variables, expresiones e instrucciones, que en su caso serán objeto de estudio en las próximas líneas. Una instrucción no es más que una acción que se realizará sobre el conjunto de datos que utiliza el programa, con el fin de transformar los datos de entrada en los de salida. En la anterior entrega de esta sección ya se describió la instrucción de asignación que permitía dotar de un valor a una variable, siendo éste el resultante de evaluar la expresión a asignar.
Aparte de esta instrucción, los lenguajes de programación llevan incorporados otras instrucciones que, a diferencia de la anterior, permiten variar el orden en que se ejecutan las instrucciones del programa. Se trata de las de bifurcación condicional e iterativas. Sin ellas, el conjunto de instrucciones que forma un programa se ejecutaría según el orden en que se encuentren dispuestas.

FIGRUA 1.



INSTRUCCIONES DE SELECCION

Una instrucción de bifurcación condicional comienza calibrando una expresión de tipo lógico (tan sólo puede tomar los valores verdadero o falso). Dependiendo de la evaluación de dicha expresión, pasará a ejecutar un conjunto de instrucciones u otro. En la figura 1 es posible ver dos diagramas de flujo, que dan idea del efecto de tales instrucciones. El rombo representa la condición asociada a la bifurcación, mientras que los rectángulos describen una o varias instrucciones. En el caso particular del formato 1, si la condición se evalúa como verdadera, se procederá a ejecutar el conjunto de instrucciones etiquetado como "bloque 1". Mientras, en el caso de que la condición se evalúe como falsa, se ejecutará el bloque de instrucciones calificado como "bloque 2". Una vez ejecutado uno de los bloques, se procede a seguir la secuencia de instrucciones, esto es, accede a ejecutar el "bloque 3". El formato 2 únicamente activa el "bloque 1" de instrucciones si la condición se evalúa como verdadera, pasando a ejecutar el "bloque 2" a continuación. En este caso no se ejecuta ninguna instrucción extra, a menos que la condición sea considerada como falsa.

Todos los lenguajes de programación utilizan una sintáxis muy férrea

La instrucción condicional por excelencia en la mayoría de los lenguajes es **if**. En el ejemplo particular de Pascal (figura 2) le acompaña la palabra reservada *then* tras la condición, indicando que las instrucciones asociadas se ejecutarán en el supuesto de que la condición se estime como verdadera. Opcionalmente puede seguirle la palabra reservada ***else***, lo que determina que las instrucciones asociadas se ejecuten si la evaluación de la condición es falsa. Hay que hacer notar que en el formato de la instrucción se encuentra incluida entre corchetes la palabra reservada else junto con su correspondiente bloque de instrucciones. Estos signos no forman parte de la instrucción. Así, los ejemplos 1 y 2 son correctos, aunque en el segundo se obvió la palabra *else*.
La combinación de condiciones con las operaciones lógicas ***or, and*** *y* ***not*** facilita la inclusión de condiciones más o menos complejas en la instrucción *if*. También resulta factible anidar ésta, de tal forma que dentro de una instrucción *if* se encuentre otra y así sucesivamente. Conviene tener en cuenta que un exceso en el anidamiento de estas instrucciones puede producir un código poco

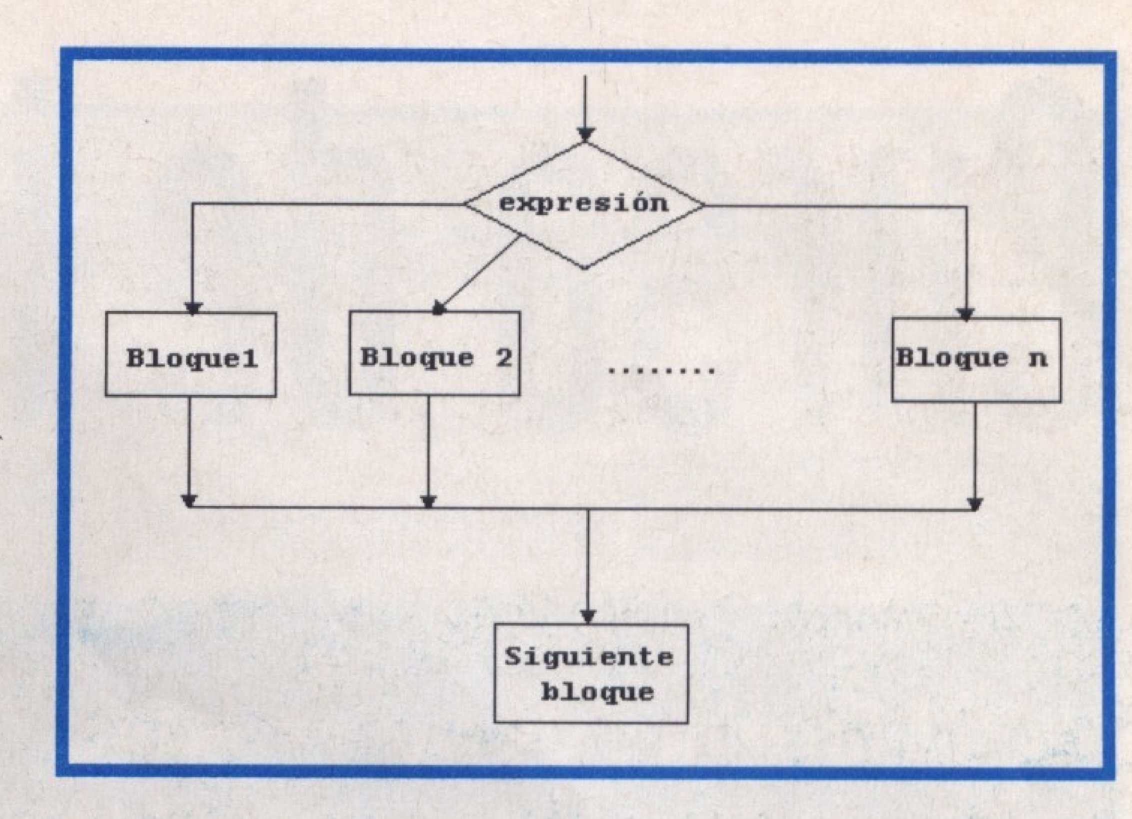
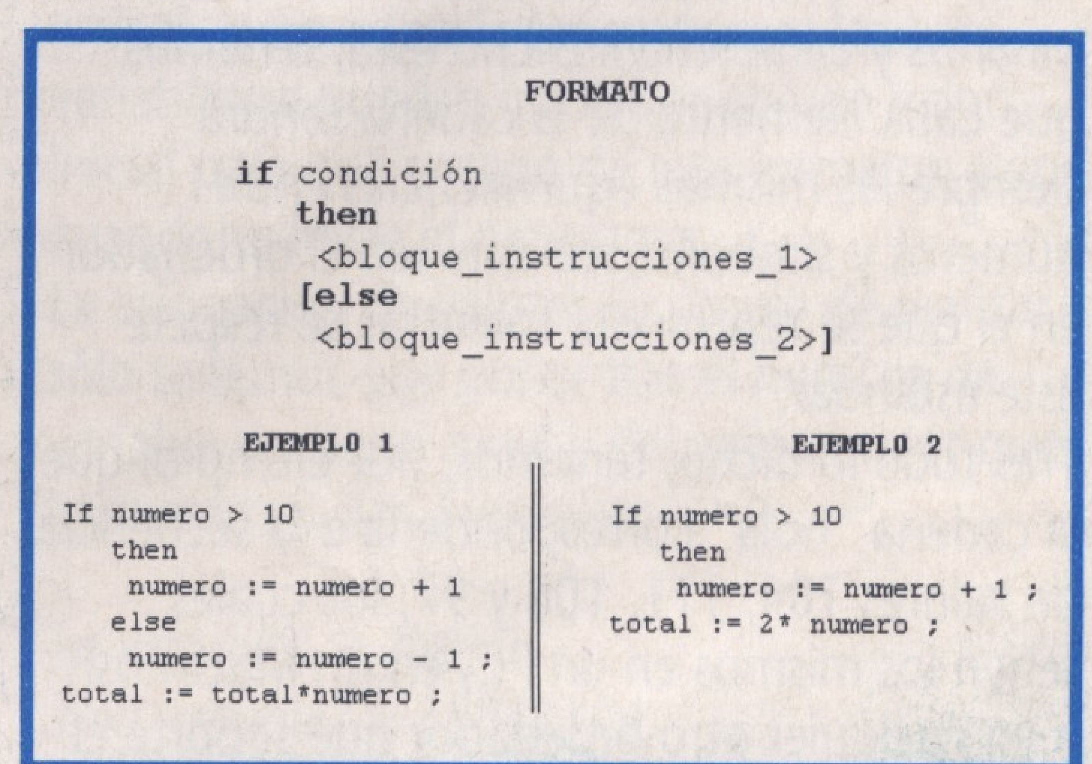


FIGURA 3.

comprensible. Debido a ello, es recomendable no anidar más de cuatro de estas instrucciones o bien utilizar una selección múltiple, cuyo funcionamiento se describe a continuación.
En primer lugar, hay situaciones en las que se desea ejecutar una serie de instrucciones dependiendo del valor de una expresión. Por ejemplo, en un menú presentado a un usuario, éste podría seleccionar entre las diversas actividades que realiza un programa. Para distinguir cada uno de los valores y hacer efectivas sus correspondientes instrucciones, serían necesarias tantas *if* anidadas como opciones presente el menú, lo que de inmediato provocaría una pérdida de claridad. Dicho obstáculo puede ser solucionado mediante una instrucción de selección múltiple, puesto que la base de las decisiones tiene lugar sobre la misma expresión (la variable que contiene la opción seleccionada por el usuario) y los valores de comprobación no pueden darse a la vez.
Esta instrucción actúa como sigue.
Inicialmente, comprueba si el valor contenido en la variable coincide con alguno de los valores asociados a las diferentes opciones de dicha instrucción. De ser así, ejecutará el conjunto de las asociadas y posteriormente pasará a hacer efectiva la siguiente instrucción a la de selección (figura 3). Normalmente estas órdenes permiten incluir una claúsula, cuya acción asociada se ejecutará en el caso de no existir ningún valor que se corresponda con el propio de la expresión asociada a la instrucción de selección múltiple, que en el caso particular de Pascal se denomina *case*.

FIGURA 2.



FORMATO

```
if condición
   then
    <bloque_instrucciones_1>
   [else
    <bloque_instrucciones_2>]
```

EJEMPLO 1

```
If numero > 10
   then
    numero := numero + 1
   else
    numero := numero - 1 ;
total := total*numero ;
```

EJEMPLO 2

```
If numero > 10
   then
    numero := numero + 1 ;
total := 2* numero ;
```

Curso de iniciación en C

Hablaremos de las variables, de su almacenamiento, y de algunos conceptos importantes del lenguaje C, así como un rápido repaso a algunos hitos de su historia.

En el anterior número de la revista comenzamos a daros los primeros conceptos acerca del lenguaje C. Hablamos de la estructura básica, de las línesas, de las funciones, de los delimitadores, la instrucción, la cadena "Hola" y los errores de compilación, y pusimos algunos ejemplos que nos eran muy útiles. En esta ocasión acaberemos de completar los conceptos básicos para meternos de lleno en el lenguaje en sí.

CÓMO SE ALMACENAN LAS VARIABLES EN MEMORIA

Cada elemento de una cadena se representa en el ordenador por convención mediante 1 solo byte (1 byte = 8 bits). Si tenemos que 1 byte puede tener poseer 256 valores diferentes (un número del 0 al 255), nos da que en un ordenador se pueden representar 256 caracteres diferentes (incluyendo letras, números y símbolos).

Todas las variables en C se declaran indicando dos campos o componentes: el tipo de dato y el nombre de la propia variable

Además de lo dicho, cada letra, número o signo está estandarizado en el llamado A.S.C.I.I. (*American Standar Code of Information Exchange*), que es un estándar aceptado mundialmente y que consta de 128 caracteres (correspondencias entre números binarios y caracteres). Dicho esto, tenemos que cada elemento de la cadena tendrá siempre las mismas equivalencias entre números y símbolos,sea cual sea el ordenador en el que se represente mientras se respete este estándar.
Tras todo lo dicho, tenemos, por ejemplo, que la cadena "hola" correspondería a la secuencia de valores 104, 111, 108 y 97, los cuales serían los mismos en un PC, en un Macintosh o en cualquier otro ordenador que soporte el estándar ASCII. De esta forma, se hace posible el intercambio de información entre ordenadores y aplicaciones.

OTROS CONCEPTOS IMPORTANTES DENTRO DEL LENGUAJE

En el lenguaje C, como en cualquier otro lenguaje, hay tres puntos que se han de tener presentes: la forma de almacenar los datos que use (las variables), la entrada y salida de los mismos y los operadores usados en éste mismo para transformarlos y combinarlos.

PERO, ¿QUÉ SON REALMENTE LAS VARIABLES?

Las variables constituyen el bloque fundamental de todos los lenguajes de programación, incluyendo el propio C. Una variable no es más que un espacio de memoria del ordenador que se ha reservado para poder almacenar en él un tipo de dato concreto, de forma que esté siempre localizable rápidamente por el programa que la utilice y que sea fácil, en consecuencia, operar con él.
Una variable puede contener diferentes valores en momentos puntuales y así, por ejemplo, si creamos un programa para calcular nóminas necesitará, al menos, poder almacenar el ratio horario (pago por hora), y las horas trabajadas del sujeto.
En este caso, por ejemplo, si se hiciera el cálculo para más de una persona con el mismo programa, tendríamos entonces que se utilizarán las mismas posiciones de memoria para almacenar los valores de cada persona para poder así calcular cada uno, por lo que dichas posiciones de memoria nos encontraríamos que son «variables» en cuanto a contenido se refiere.

DECLARACIÓN DE UNA VARIABLE

Todas las variables en C se declaran indicando dos campos o componentes: el tipo de dato y el nombre de la propia variable.
Cuando se declara una variable, el compilador lo que hace es reservar una cantidad apropiada de memoria para almacenarla. Pongamos un ejemplo:
Int numero1;

Hay varios tipos de variables, de forma que disponemos de las clases más adecuadas para cada situación

En este caso, lo que hemos hecho es definir una variable del tipo *int*, que es una abreviación de *integer* en inglés (traducido como *entero*), a la cual se le ha dado el nombre de *numero1*. Al ser una variable tipo *entero*, lo que hará el compilador será reservar un espacio de memoria de 2 bytes (16 bits) para que se pueda almacenar en ellos un dato con valores posibles comprendidos entre –32768 y +34767.

PALABRAS FINALES

Con lo explicado a lo largo de este capítulo, que nos ha ocupado casi dos entregas de la revista, se pueden dar por explicados los conceptos más elementales relacionados con el lenguaje C, de modo que podemos empezar a profundizar en él. En el próximo capítulo se profundizará, entre otras cosas, el tema de las variables, fundamentales de entender y saber usar.

UNA NUEVA FASE EN LA PROGRAMACION EN C

En el anterior capítulo del presente curso de lenguaje C quedaron expuestos todos los conceptos básicos relativos a este lenguaje de programación, explicando qué es un compilador y para qué sirve, haciendo un poco de historia sobre los orígenes de este popular lenguaje e incluso dando un primer ejemplo de programa realizado en C para que el lector pudiera ver cómo se escribe en él.
Ahora pasamos a detallar partes más concretas de este lenguaje, que pueden considerarse básicas en programación.
Para empezar, explicaremos todos los tipos de variables que existen en C, explicando para qué sirve cada una y sus limitaciones. Después estudiaremos el uso de las instrucciones más básicas de este lenguaje, que son las llamadas

de Entrada/Salida de datos, las que permiten al programa que nos muestre información en pantalla y, a su vez, pueda recibir datos del usuario durante la propia ejecución.

TIPOS DE VARIABLES

Como es lógico, hay varios tipos de variables, de forma que disponemos de las clases más adecuadas para cada situación. A continuación se detallan todos los tipos empezando por el char:

- char: es la única que no se usa para almacenar números propiamente dichos, sino que está hecha para que se guarden en ella caracteres. Posee un tamaño de 1 byte (8 bits), por lo que se pueden representar 256 caracteres, que son los que tenemos en la tabla ASCII.

A las variables se les puede dar un valor inicial al definirlas

Además del tipo char existen también tres tipos de variables que se usan para almacenar enteros de diversos tamaños, o sea, números que no pueden poseer decimales:

- short: posee 16 de tamaño, 2 bytes, y se usa para almacenar números que estén comprendidos entre -32768 y 32767.
- int: tiene 4 bytes de tamaño. Se usa para almacenar valores comprendidos entre -2gb y +2gb.
- long: es el tipo de entero más largo que se puede definir. Posee 8 bytes de tamaño y puede almacenar números gigantescos (-2^63 hasta +2^63).

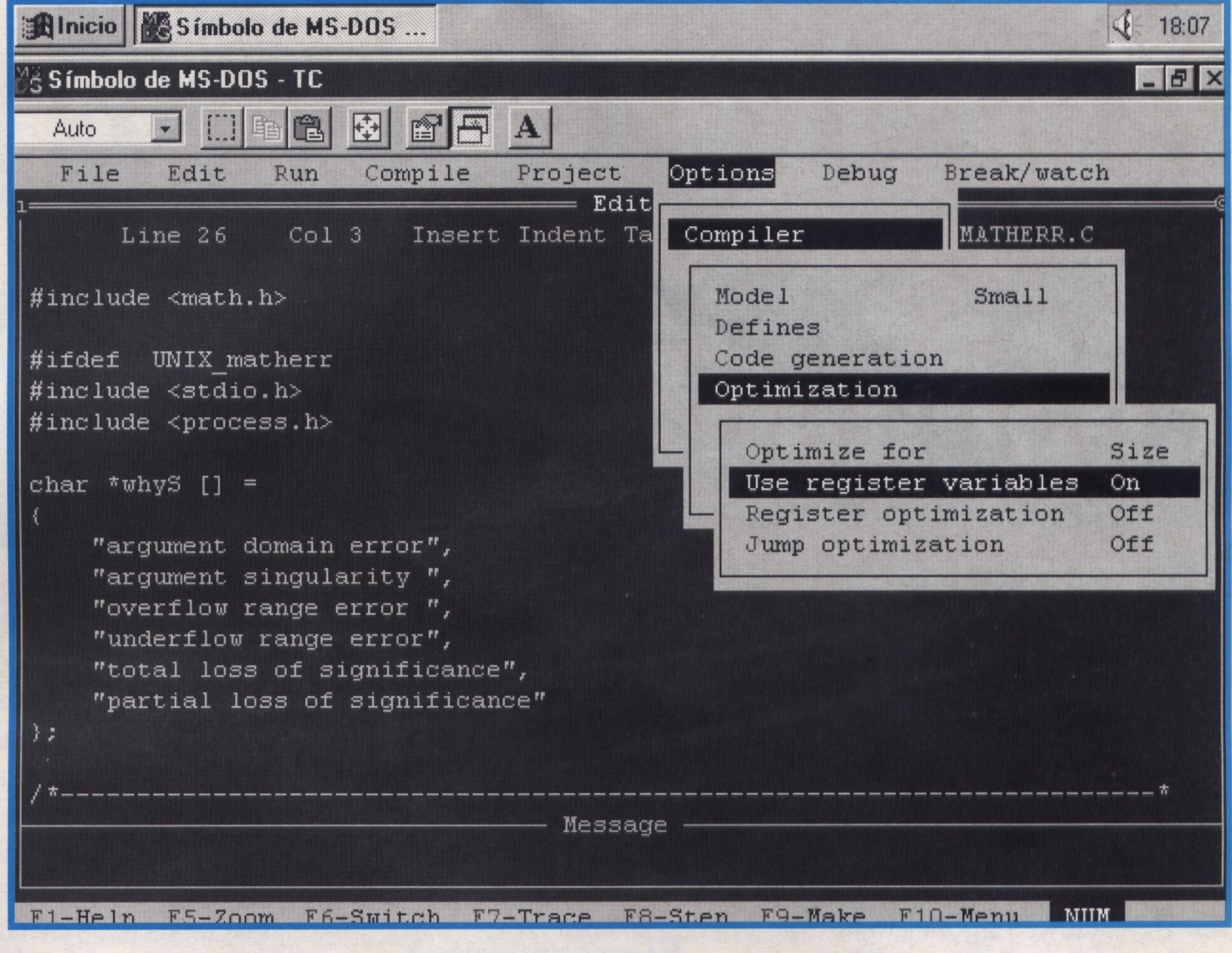
Por último tenemos los tipos de coma flotante, que existen en dos tamaños y permiten poder almacenar (manejar) números que posean decimales. Son lo siguientes:

- float: el más pequeño, posee 4 bytes de tamaño, permite poder almacenar valores comprendidos entre +1.175494e^-38 y +3.402823^38.
- double: de un tamaño de 8 bits, como el tipo long entero, y puede almacenar valores comprendidos entre 2.22^-308 y 2.22^+308.

Algunos ejemplos de definición de variables son los siguientes:

char LETRA;
int NUMERO;
float REAL;

Además de estos diferentes tipos de variable, tanto char como las diversas versiones de enteros (short, int y long) poseen además la posibilidad de incluir un prefijo de definición



EL CLASICO TURBO C.

para poder definir los mismos tipos pero sin signo. Para ello tan sólo tenemos que poner el prefijo *unsigned* delante de la definición del propio tipo de variable.

Las funciones de entrada/salida son las más importantes de cualquier lenguaje

Ejemplos:

unsigned char Variable1;
unsigned int Variable1;
unsigned short Variable2;

Definiendo una variable como un tipo de dato unsigned, pierde la capacidad de poder contener valores negativos, aunque en contrapartida a su vez se duplica el rango de valores positivos que podemos almacenar en ella. Dicho lo anterior, tenemos que un short que en definición normal sólo puede albergar valores del *-32768 al + 32767*, al definirlo como unsigned el rango pasa a ser desde 0 a 65535.
El uso de variables con o sin signo dependerá del uso que queramos dar en cada caso de la misma, así como del rango de valores que necesitemos poder representar.
Otra forma permitida de definir variables que deja ahorrar espacio de texto es la de indicar varias definiciones en una sola línea, con la única condición, claro está, de que todas las variables sean del mismo tipo. Ejemplo de esta clase de definición sería:

int NUMERO, NUMERO1, NUMERO2, NUMERO3;
float REAL, NUM1, NUM2, TEMP, COOL;

El ahorro de espacio para definir las variables queda visible rápidamente, ya que sólo estas dos líneas de ejemplo hubiesen necesitado nueve líneas de código para realizar la definición de las mismas de forma individual.

INICIALIZACION DE VARIABLES

Como ya se explicó en el anterior capítulo de la serie, las variables se definen escribiendo el tipo que queremos definir (uno de los expuestos anteriormente) seguido del nombre de la propia variable que nos interesa en ese momento.
Hasta aquí es fácil de entender (como se puede comprobar en los ejemplos dados más arriba), pero además tenemos que las variables se pueden inicializar con un valor inicial para que posean de esta forma un valor cuando se inicie el programa.
La asignación de valores es simple de realizar, sólo tenemos que poner tras el nombre de variable un signo igual y después el valor que le queramos dar. Ejemplos:

int variable1 = 200;
long variable2 = 35042304;
float variable3 = 2344.33;

Primeros pasos en el ensamblador

Antes de iniciarnos en la programación en ensamblador, explicaremos los conceptos y elementos fundamentales del hardware de nuestro PC.

¿QUÉ SE NECESITA PARA SEGUIR EL CURSO?

Para poder practicar todo lo que se enseñará sobre el 8086, el lector necesitará, además del propio ordenador PC (algo obvio), tener instalado el sistema operativo DOS (MS-DOS, PC-DOS...) o uno que sea compatible con él (Windows 95 por ejemplo puede ejecutar aplicaciones DOS). Si tenemos lo anterior, sólo falta disponer de una herramienta que se llama ensamblador o *assembler* (igual que el propio lenguaje) y que es la que nos permite crear programas con extensión *.EXE* codificadas en binario, formato que es el que se usa en DOS para los programas.

Hay varias herramientas ensamblador de diversos fabricantes, y todas, en general, son productos de calidad. De la casa Microsoft tenemos el Macroassembler, y de la casa Borland, por ejemplo, el Turbo Assembler. Ambos han sido y son muy usados. A elección del lector queda cuál conseguir y usar.

¿PARA QUÉ SIRVE EL ENSAMBLADOR?

Cuando un programador escribe un programa en ensamblador, lo que hace es escribir un fichero de texto (un documento) en el que cada línea (aproximadamente) es una instrucción del procesador (ensamblador). Pero el ordenador no entiende de textos, no puede procesar un documento, ya que es binario y sólo sabe ejecutar las instrucciones codificadas en binario, y he aquí donde interviene la herramienta de *Assembler*: Una vez tenemos nuestro programa escrito, el ensamblador coge el texto (llamado en informática *fuente*) y convierte las instrucciones escritas en texto a bytes que corresponden a las mismas en formato binario reconocible por la C.P.U.

El ensamblador convierte las instrucciones escritas en texto a bytes

Una vez codificadas todas las líneas del *fuente*, el *Assembler* las guarda todas en un fichero binario con extensión *.EXE* (por ejemplo *PROGRAM.EXE*)

Con lo explicado, tenemos que cuando ejecutamos un programa *.EXE*, el procesador puede leer su contenido directamente, ya que son instrucciones ensamblador en formato binario (lo que se conoce por código máquina) interpretable por la C.P.U.

PROFUNDIZANDO EN EL 8086

Antes de empezar a detallar nada sobre el propio ensamblador, sería bueno que el lector supiera más cosas sobre el propio microprocesador 8086.

Desde que aparecieron los primeros chips, el mercado de los procesadores se dividió en dos variantes, dos tecnologías diferentes de diseño. Son las llamadas tecnologías R.I.S.C. y C.I.S.C.:

Los microprocesadores diseñados con la tecnología R.I.S.C., siglas que provienen del inglés *Reduced Instruction Set Computer*, se caracterizan por poseer sólo unas pocas instrucciones básicas y con la particularidad de que todas se ejecutan normalmente en un solo ciclo de oscilador. Así pues, tenemos que estos chips ejecutan los programas muy rápido pero que requieren muchas instrucciones ensamblador para realizar operaciones.

Por otro lado tenemos los procesadores diseñados con la tecnología C.I.S.C., que proviene de las siglas *Complex Instruction Set Computer*. Estos chips poseen un diseño interno complejo y se caracterizan porque sus instrucciones son muchas y pueden realizar operaciones complejas. Como contrapartida tienen el defecto consecuente de su complejidad; casi todas las instrucciones tardan un mínimo de dos ciclos de reloj en ejecutarse y un máximo más o menos indefinido (una multiplicación puede tardar cientos de ciclos).

Los procesadores se han dividido en dos tecnologías: Cisc y Risc

El 8086 es un chip basado en esta tecnología, con lo que posee instrucciones muy potentes pero con una velocidad de ejecución no muy elevada.

Para acabar el tema de las dos tecnologías de microprocesadores, decir que mientras en los últimos años los R.I.S.C. perdieron terreno y se popularizaron los C.I.S.C. (PC es el mejor ejemplo de la expansión de los chips C.I.S.C. en el mercado de hardware), decir que el sucesor de los actuales chips instalados en PC, que se llama IA-64, está siendo diseñado usando una tecnología derivada del R.I.S.C. (una evolución de la misma), por lo que se puede decir que habrá un resurgimiento de esta tecnología, mientras que la C.I.S.C. pasará a mejor vida (al menos en el mundo del PC).

¿QUÉ PUEDE HACER UN 8086?

El set de instrucciones que posee el 8086 puede clasificarse en siete grupos de utilidad:

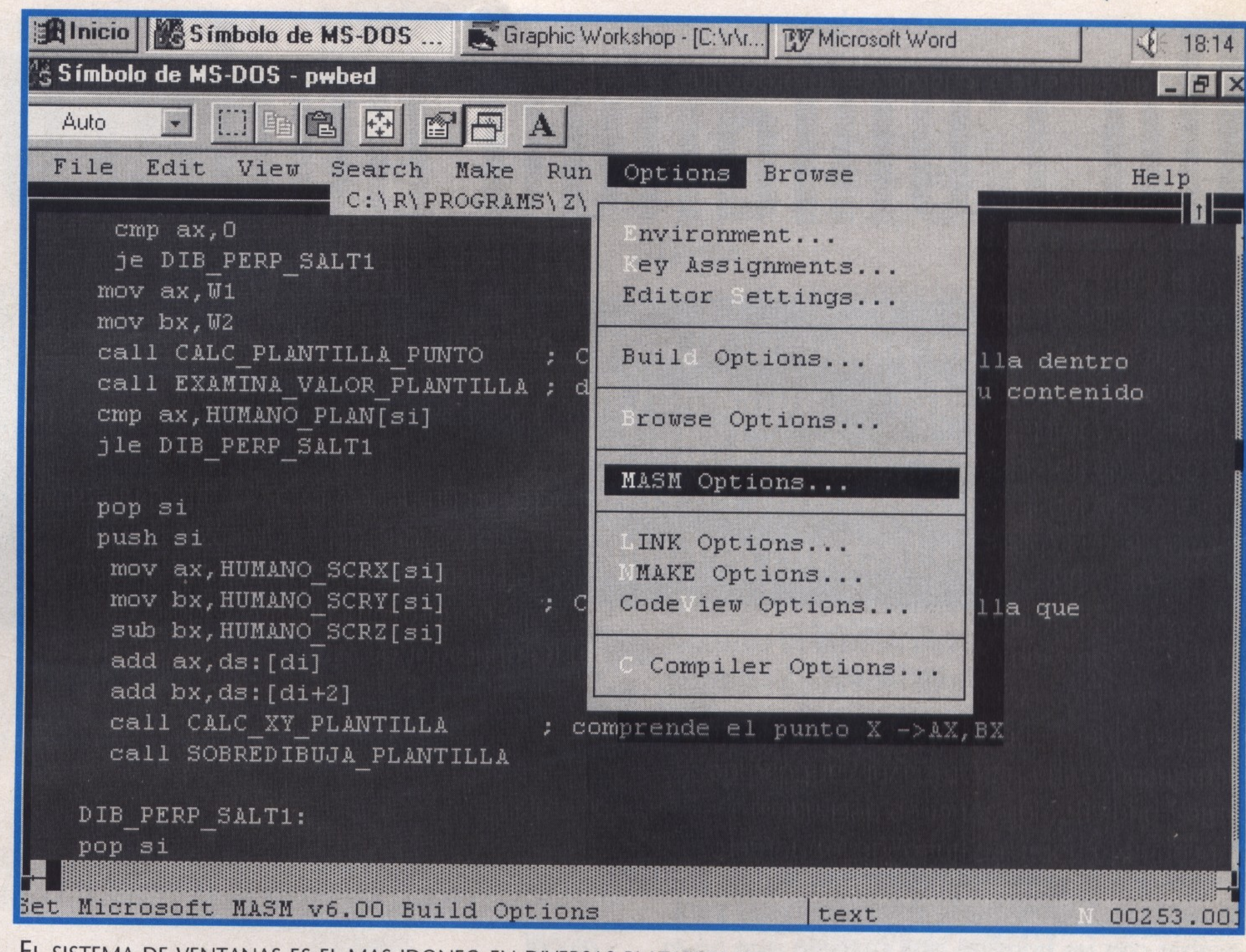
- Instrucciones de transferencia de datos.
- Instrucciones aritméticas.
- Instrucciones de manejo de bits.
- Instrucciones de transferencia de control.
- Instrucciones de manejo de cadenas.
- Instrucciones de interrupción.
- Instrucciones de control del microprocesador.

RESUMEN DE CONCEPTOS

En este primer capítulo de la serie, que acaba con estas últimas notas, se han explicado con detalle todos los conceptos fundamentales relacionados con el hardware del PC, en qué consiste exactamente el ensamblador y cómo se puede programar con él. Ahora que ya se ha situado convenientemente el lector, en los siguientes apartados se empezarán a detallar aspectos más técnicos de la arquitectura interna del chip.

LA MEMORIA Y LOS TIPOS DE DATOS QUE MANEJA EL PROCESADOR

Saber con qué clases de datos puede operar el microprocesador y cómo se almacenan físicamente en la memoria es vital para llegar a poder programar en Ensamblador correctamente. En este capítulo del curso de Ensamblador trataremos todos los conceptos básicos que estén relacionados con la memoria del ordenador, así como los tipos de datos que son soportados por el chip 8086, la forma en que se almacenan las variables en memoria y otros datos de interés que están relacionados con ello. El capítulo nos ocupará más de lo que por el momento nos entra dentro de este número, pero esperamos poder completarlo en la siguiente entrega de la revista. Todos los aspectos del ensamblador deben conocerse bien para poder manejar este entorno, pero en especial el de la memoria y los tipos de datos que puede utilizar el microprocesador. Por ello miraremos este apartado con detenimiento, con objeto de que no nos quede ningún cabo suelto.



El sistema de ventanas es el mas idoneo en diversas plataformas.

LA FORMA DE LA MEMORIA Y LA INFORMACIÓN

Como ya debe saber el lector, la memoria del ordenador se compone de unidades de almacenamiento llamadas bits, las cuales tienen dos estados posibles: 0 y 1 (electricidad o no electricidad) y, por lo tanto, sirven para almacenar números en formato binario.

Toda la memoria está dividida en celdas de 8 bits llamadas bytes

A partir de lo dicho es fácil entender que todo lo que se tiene almacenado en un ordenador en un momento determinado (programas, datos, gráficos, textos...) no son más que números binarios almacenados en los bits que conforman la memoria.

EL ELEMENTO BYTE DE LA MEMORIA DEL ORDENADOR

Los bits de memoria no son accesibles individualmente por el microprocesador, estando agrupados en celdas (bloques) de ocho bits (octetos de bits). A cada uno de los octetos de bits en los que se divide la memoria se le llama byte.
El byte, como es sabido por la mayor parte de usuarios de informática, es la unidad mínima de memoria que se puede usar en un ordenador (dejando de lado, a estas alturas, los bites), y a toda la memoria se accede por bytes, los cuales se hallan localizables secuencialmente, como se explicó en el anterior capítulo, mediante un número de dirección física que corresponde a la posición que ocupa dentro de la cadena secuencial, teniendo así el byte de memoria 0, el byte 1, el byte 2, etc.

LAS MAGNITUDES QUE MANEJA LA MEMORIA

Las magnitudes para medir cantidades de memoria son un poco *originales*, por decirlo de alguna manera, de forma que para llegar a la cantidad de 1 Kilobyte no son necesarios 1.000 bytes, como podría pensarse (y sucede en todos los sistemas de medida), sino que son necesarios 1.024 bytes. Esto sirve también para cantidades mayores, a medida que vamos subiendo de magnitud.
Todo el resto de magnitudes funciona de forma idéntica, y así resulta que para tener 1 Megabyte son necesarios 1.024 Kilobytes, y para obtener 1 Gigabyte se necesitan a su vez 1.024 Megabytes.
Dicho todo lo anterior, en el caso de que alguien dijera que tiene, por ejemplo, un computador con 32 Megabytes de memoria RAM, haciendo la multiplicación 32 * 1024 * 1024, obtendríamos la cantidad de memoria que posee en bytes (33.554.432 bytes). A su vez, recordemos que 1 byte son, por su parte, 8 bits, y por lo tanto, multiplicando el resultado por 8 se obtiene el número de bits que posee (o sea, un número gigantesco que no vale la pena calcular).

FUNCIONAMIENTO DE UN BYTE

Los bits de un byte se numeran de derecha a izquierda (al igual que en decimal el dígito de menor peso o valor es el que está situado más a la derecha) y por equivalencia matemática se obtiene que cada uno de los bits que tenemos corresponde a una potencia de 2.

Los bits se numeran de derecha a izquierda y su equivalencia matemática es de potencia 2

Dicho esto, es fácil convertir un número que tengamos en formato binario al formato decimal. Pongamos, por ejemplo, el número 01111101b. El número decimal que le corresponde se obtiene a partir de sumar las potencias de dos de todos los bits que están a 1, siendo los exponentes de cada uno de ellos el número de posición que ocupa respecto a la derecha.
Todo lo explicado hata aquí se resumiría en la siguiente operación:

*Decimal = (0*2^7) + (1*2^6) + (1*2^5) + (1*2^4) + (1*2^3) + (1*2^2) + (0*2^1) + (1*2^0).*

Con esto queda demostrado además (si no, haga el cálculo) que con ocho bits se pueden representar números comprendidos entre 0 y 255 (2^8= 255).

Autor: David Ferriz

Un comando poderoso

No es el mejor juego, ni el más perfecto, pero es un juego original. En efecto, quizá nos han llegado aquí programas algo mejores, pero todos ellos estaban demasiado directamente basados en clásicos de antaño; son juegos efectivos por poco arriesgados.

Tres pollitos se encuentran bajo tu custodia en el programa. De las alturas irá cayendo una serie de bombas y tu misión es resguardar a los pollitos de ellas. Para pasar de fase, basta con que mantengas con vida a uno de ellos. Como defensa, cuentas con una mirilla que se puede desplazar a tu antojo por la pantalla. Tienes municiones ilimitadas para acabar con las bombas que caen, lo que no es nada fácil cuando vienen muchas a la vez. Por lo demás, los pollitos están distribuidos de diversa forma en el suelo, y se moverán a izquierda o derecha al tiempo que lo hace el personaje que tú manejas.

A continuación transcribimos algunos comentarios del autor. "Para hacer *Comando Pelota* tardamos cerca de dos meses y medio. Primero empecé yo creando los modelos 3D y las animaciones de las pelotas para meterlas en un juego, sin ninguna idea en especial. Después ideamos una mecánica para el juego, y mientras, mi hermano fue creando enemigos y yo hacia los escenarios y el programa."

Es una curiosa forma de trabajar en equipo, pero se aproxima de alguna forma a la manera profesional de elaborar un juego. Aunque, desde luego, en un gran proyecto participa mucha más gente. "La mayoría de los procesos del juego están controlados por un marcador de tiempo llamado marca, que me ha servido de gran utilidad para hacer el cambio de nivel, iniciar unos enemigos y eliminar otros, etc." Os mostramos a continuación el código principal del juego, que nos han proporcionado los autores. Encontraréis de cuando en cuando breves anotaciones que indican para qué sirve cada listado.

ESTE EL NOMBRE QUE HAN DADO A SU SELLO.

```
program COMANDO_PELOTA;
global
fuente_2;
puntos=0;
energia=100;
energiaboss=300;
bichos=3;
marca=0;
marca2=0;
s_disparo;
s_explo;
s_explo2;
s_uh;
s_ouh;
s_iaia;
s_explob;
s_gritob;
s_au;
S_ARG;
begin
set_fps(22,0);
```

No es un alarde de técnica, pero sí uno de los programas más originales hechos en DIV

A continuación se pone el programa a 22 frames por segundo y se inician las intros.

```
start_fli("fli\devilish.flc",0,0);
while(frame_fli()<>0 and scan_code==0 and not
mouse.left)
frame;
end
end_fli();
fade_off();
fade_on();
```

DESDE AQUI NO DEJARAN DE CAER BOMBAS...

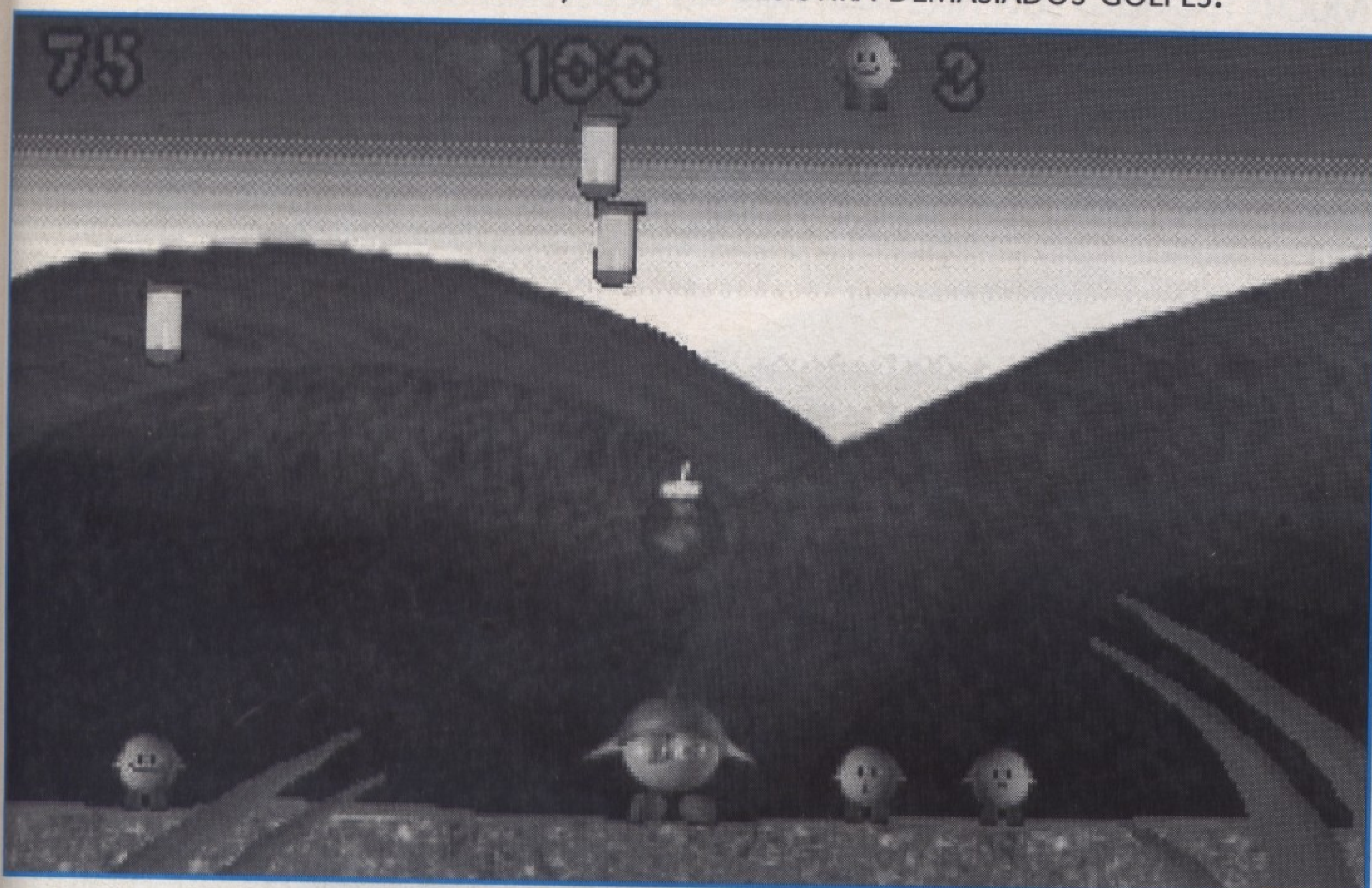


```
SET_MODE(M640X480);
sTART_FLI("FLI\cOMANDO.FLc",0,0);
WHILE( SCAN_CODE==0 AND NOT
MOUSE.LEFT)
frame_fli();
FRAME;
END
END_FLI();
FADE_OFF();
FADE_ON();
set_mode(m320x200);
set_fps(22,0);
```

Después se ponen en modo de pantalla los frames y se cargan la paleta, el fichero, las fuentes y los sonidos.

```
load_pal("pal\pal.pal");
load_fpg("cp.fpg");
FUENTE_2=LOAD_FNT("v2.FNT");
s_disparo=load_pcm("6DISPA.pcm",0);
s_explo2=load_pcm("boom.pcm",0);
s_explo=load_pcm("exp.pcm",0);
s_uh=load_pcm("uh.pcm",0);
s_ouh=load_pcm("grito2.pcm",0);
s_iaia=load_pcm("grito1.pcm",0);
s_explob=load_pcm("expBOSS.pcm",0);
s_gritob=load_pcm("grito.pcm",0);
s_au=load_pcm("plof.pcm",0);
S_ARG=LOAD_PCM("AAAA.PCM",0);
WRITE_INT(FUENTE_2,10,0,0,&PUNTOS);
WRITE_INT(FUENTE_2,215,0,0,&bichos);
WRITE_INT(FUENTE_2,120,0,0,&energia);
heli();
heli2();
heli3();
heli4();
malo();
malo1();
malo2();
malo3();
malo4();
```

EL PROTAGONISTA LLEVA UN CASCO, PERO NO RESISTIRA DEMASIADOS GOLPES.



```
helil();
helil2();
fin();
finalboss();
suelo();
peques();
peques1();
peques2();
personaje();
manza();
manza2();
bomba();
bomba2();
bomba3();
bomba4();
hojas();
mirilla();
NIVEL1();
nivel2();
nivel3();
nivel4();
NIVELBOSS();
CORAZON();
marcapeque();
GAMEOVER();
```

Luego tenemos que mostrar los procesos en pantalla.

```
put_screen(0,13);
loop frame;
```

Conseguimos que los enemigos no paren de salir, controlando el número, la velocidad, el ángulo y el lugar.

```
IF(RAND(0,900)<30)
misil(RAND(10,310),RAND(0,0),RA
ND(3,2));
end
if(rand(0,1300)<30)
bolapin(RAND(0,320),RAND(-
5,5),RAND(2,2));
end
if(rand(0,1600)<30)
misilper(rand(10,310),rand(0,0),ra
nd(2,2));
end
if(rand(0,1300)<30)
finalene(rand(10,310),rand(0,0),ra
nd(2,5));
gota(rand(10,310),rand(0,0),rand(
4,2));
bolaazul(rand(10,310),rand(-
5,5),rand(4,2));
end
if(energiaboss<1);marca=40000;
end
IF(KEY(_D)AND(KEY(_A)AND(KEY(_
V)AND(KEY(_I)AND(KEY(_D))))));en
ergia=500;
end
IF(KEY(_g)AND(KEY(_o)AND(KEY(_
d))));bichos=4;
end
if(key(_l)and(key(_e)and(key(_v)an
d(key(_2)and(marca<2500)))));ma
rca=2500;
end
if(key(_l)and(key(_e)and(key(_v)an
d(key(_3)and(marca<5000)))));ma
rca=5000;
end
if(key(_l)and(key(_e)and(key(_v)an
d(key(_4)and(marca<7500)))));ma
rca=7500;
end
if(key(_l)and(key(_e)and(key(_v)an
d(key(_5)and(marca<10000)))));m
arca=10000; end
```

Si se pulsan las diferentes combinaciones de teclas cambia de nivel y sube la energía. Si se pulsa ESC, se sale del juego y aparecen los créditos.

```
IF(KEY (_ESC));
fade(20,40,200,2);
WHILE(FADING)FRAME; END
EXIT("-PROGRAMACION Y DISE¥O:
DAVID FERRIZ - DISE¥O,INTRO Y
SONIDO: FCO. FERRIZ-
-DEVILISH GAMES 1998/99- -TELF-
96 580 65 38- VILLENA
(ALICANTE)-",0);
end
IF(ENERGIA<1);
FADE(200,70,200,1);
WHILE(FADING)FRAME; END
EXIT("GAME OVER",0);
end
```

Hay que revisar, uno a uno, todos los parámetros para que el programa funcione

Si el marcador de energía es menor de 1, entonces sales del juego con un fundido de pantalla.

```
if(bichos<1);
FADE(200,0,0,1);
WHILE(FADING)FRAME; END
EXIT("GAME OVER",0);
```

Si no queda con vida ninguna de las tres pelotitas se sale del juego con un fundido.

```
end
end
```

El proceso *process personaje()* controla todas las acciones del personaje principal.

```
process personaje()
begin
x=160; y=170;
graph=2;
REPEAT          frame;
```

Si colisionas con algún enemigo se te restan 3 puntos de energía.

```
if(collision(type misil)or
collision(type heli)or collision(type
heli2)or collision(type helil2)or
collision(type heli3)or collision(type
malo)or collision(type heli4)
or collision(type malo1)or
collision(type malo2)
or collision(type malo3)or
collision(type malo4)
or collision(type helil)or
collision(type bolapin)
or collision(type misilper)or
collision(type gota)
or collision(type finalene)or
collision(type bolaazul))
energia-=3;graph=16;frame;
sound(s_ouh,50,256);
end
```

CON QUE SOBREVIVA UNA DE LAS PELOTITAS HABRAS PASADO DE FASE.



```
end
marca+=1;
```

El marcador que hemos dado en llamar, en un alarde de originalidad, *Marca* se encarga de sumar 1 en cada uno de los frames, para que cuando llega a cierto número se ejecuten unas cosas y se eliminen otras.

```
if(key(_right)and x<300 and
not(key(_down)and
not(key(_left))))flags=0;x+=4;
graph++;
if( graph>11);graph=2;
end
end
if(key(_left)and x>10 and
not(key(_down)and
not(key(_right))))flags=1;x-=4;
```

```
graph++;      end
if( graph>11);graph=2;
end
if(not(key(_left)or(key(_right))))gra
ph=1;
end
if(marca>2500);put_screen(0,42);
```

Cuando el marcador *Marca* llega hasta 2.500, pone de fondo el gráfico del nivel 2.

```
end
if(marca>5000);put_screen(0,106);
```

Cuando *Marca* llega a 5.000, pone de fondo el gráfico del nivel 3.

```
end
if(marca>7500);put_screen(0,107);
```

Cuando Marca llega a 7.500 pone de fondo el gráfico del nivel 4.

```
end
if(marca>9999);put_screen(0,32);
```

Al variar la configuración del marcador el juego irá avanzando

Cuando llega a 9.999, pone el gráfico del nivel del monstruo.

```
end
if(marca>40000);
put_screen(0,33);y=400;
end
UNTIL(ENERGIA<1 or
bichos<1);FROM GRAPH=88 TO
93;FRAME;
```

Elimina al protagonista cuando no queda energía o bichos vivos.

```
end
end
```

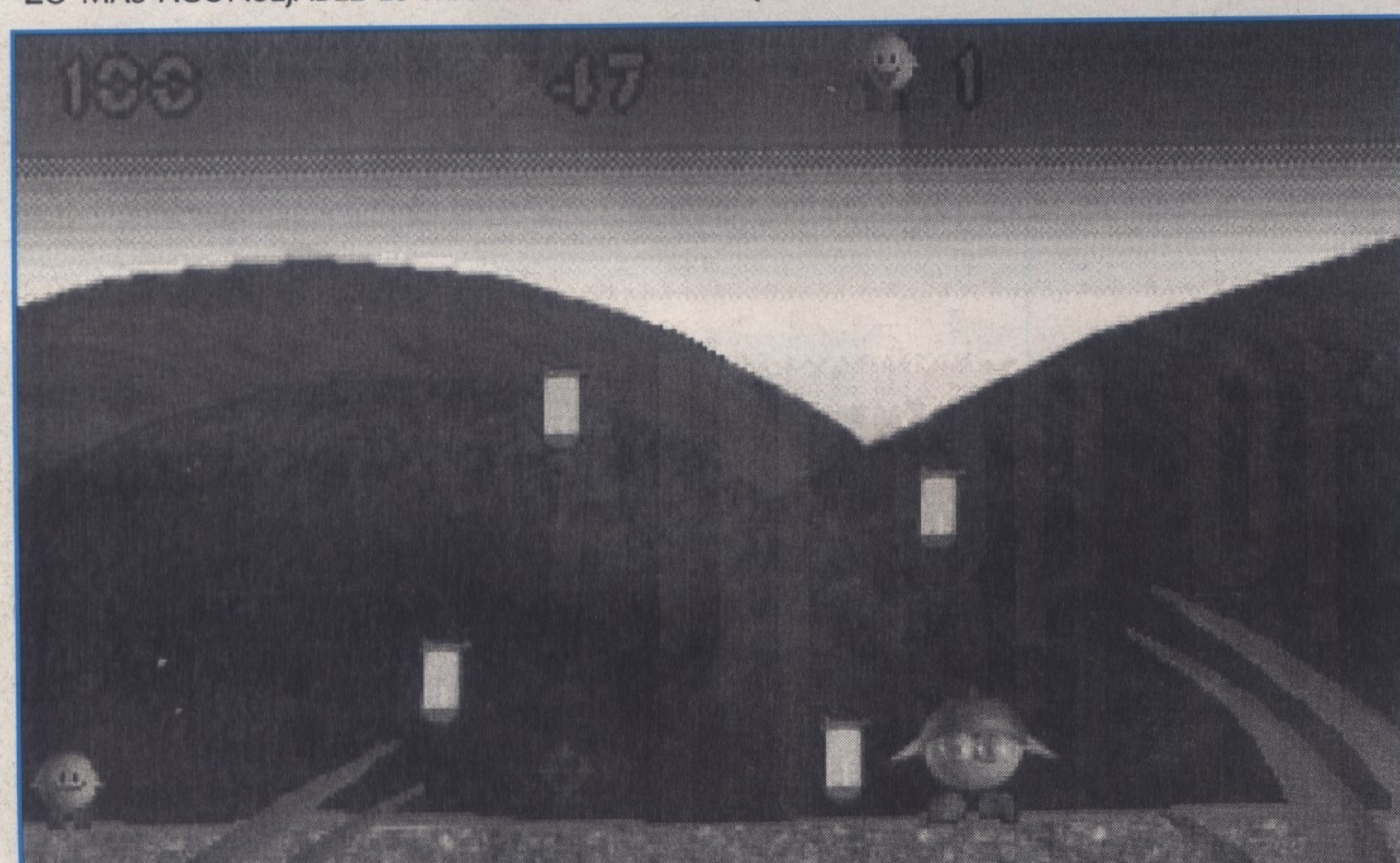
Maneja la mirilla del protagonista.

```
process mirilla();
begin
x=160;
y=100;
graph=75;
REPEAT
frame;
if(key(_right)and x<305 and
not(key(_left)))
x+=9;
end
if(key(_left)and x>10 and
not(key(_right)))
x-=9;
end
if(key(_up)and y>10 and
not(key(_down)))
y-=9;
end
if(key(_down)and y<170 and
not(key(_up)))
y+=9;
end
if(key(_space))graph++;if(graph>7
5)
graph=72;
sound(s_disparo,100,256);
```

Mientras se pulsa la tecla *space*, pone el gráfico de disparo y ejecuta el sonido del disparo.

```
end
end
if(not key(_space))
graph=72;
```

LO MAS ACONSEJABLE ES MANTENER VIVO AL QUE ESTÉ CERCA DE TI.



MUCHA HABILIDAD SERA NECESARIA PARA NO ACOSTUMBRARSE A ESTA PANTALLA.



Si no se pulsa space, tiene el gráfico de mirilla.

```
end
if(marca>40000);break;
end
UNTIL(ENERGIA<1);
END
```

El proceso *process misil()* controla los misiles enemigos.

```
PROCESS misil(X,INC_X,INC_Y)
BEGIN
if(marca<3500);
GRAPH=17;
Y=-20;
REPEAT
X+=INC_X;
Y+=INC_Y;
FRAME;
gph++;
if(graph>21)graph=17;
end
if(collision(type suelo));break;
end
if(collision(type bomba)or
collision(type bomba2)or
collision(type bomba3)or
collision(type bomba4));break;
end
until(collision(type
mirilla)and(key(_space)))puntos+=
25;FROM GRAPH=164 TO
173;FRAME;
end
if(graph>164)
sound(s_explo,50,256);
end
end
end
```

El proceso *process gota()* controla los enemigos en forma de gota.

```
PROCESS gota(X,INC_X,INC_Y)
BEGIN
if(marca>7500);
if(marca<9900);
GRAPH=268;
size=70;
Y=-20;
REPEAT
X+=INC_X;
Y+=INC_Y;
FRAME;
graph++;
if(graph>276)graph=268;
end
if(collision(type suelo));break;
end
if(collision(type bomba)or
collision(type bomba2)or
collision(type bomba3)or
collision(type bomba4));break;
end
until(collision(type
mirilla)and(key(_space)))size=100;
puntos+=25;FROM GRAPH=213
TO 222;FRAME;
enD
If(graph>213)
sound(s_ARG,60,100);
end
end
end
end
PROCESS finalene(X,INC_X,INC_Y)
```

Enemigos de la pantalla del final, el Boss de la fase.

```
BEGIN
if(marca>10000);
if(energiaboss>15);
GRAPH=299;
Y=-20;
REPEAT
X+=INC_X;
```

Juegos ganadores: 1º

```
Y+=INC_Y;
FRAME;
graph++;
if(graph>307)graph=299;
end
if(collision(type suelo));break;
end
if(collision(type bomba)or
collision(type bomba2)or
collision(type bomba3)or
collision(type bomba4));break;
end
until(collision(type
mirilla)and(key(_space))or
marca>40000)puntos+=25;FROM
GRAPH=308 TO 315;FRAME;
end
if(graph>309)
sound(s_explo,120,300);
end
end
end
end
PROCESS bolaazul(X,INC_X,INC_Y)
```

El apartado de los enemigos es uno de los que tenemos que realizar con más cuidado

Para introducir los enemigos del cuarto nivel:

```
BEGIN
if(marca>7500);
if(marca<10000);
GRAPH=244;
Y=-20;
REPEAT
X+=INC_X;
Y+=INC_Y;
FRAME;
if(x<0);x=320;
end
if(x>320);x=0;
end
graph++;
if(graph>251)graph=244;
end
if(collision(type suelo));break;
end
if(collision(type bomba)or
collision(type bomba2)or
collision(type bomba3)or
collision(type bomba4));break;
end
until(collision(type
mirilla)and(key(_space)))puntos+=
25;FROM GRAPH=290 TO
298;FRAME;
end
if(graph>295)
sound(s_EXPLO,50,256);
end
end
end
end
PROCESS bolapin(X,INC_X,INC_Y)
BEGIN
```

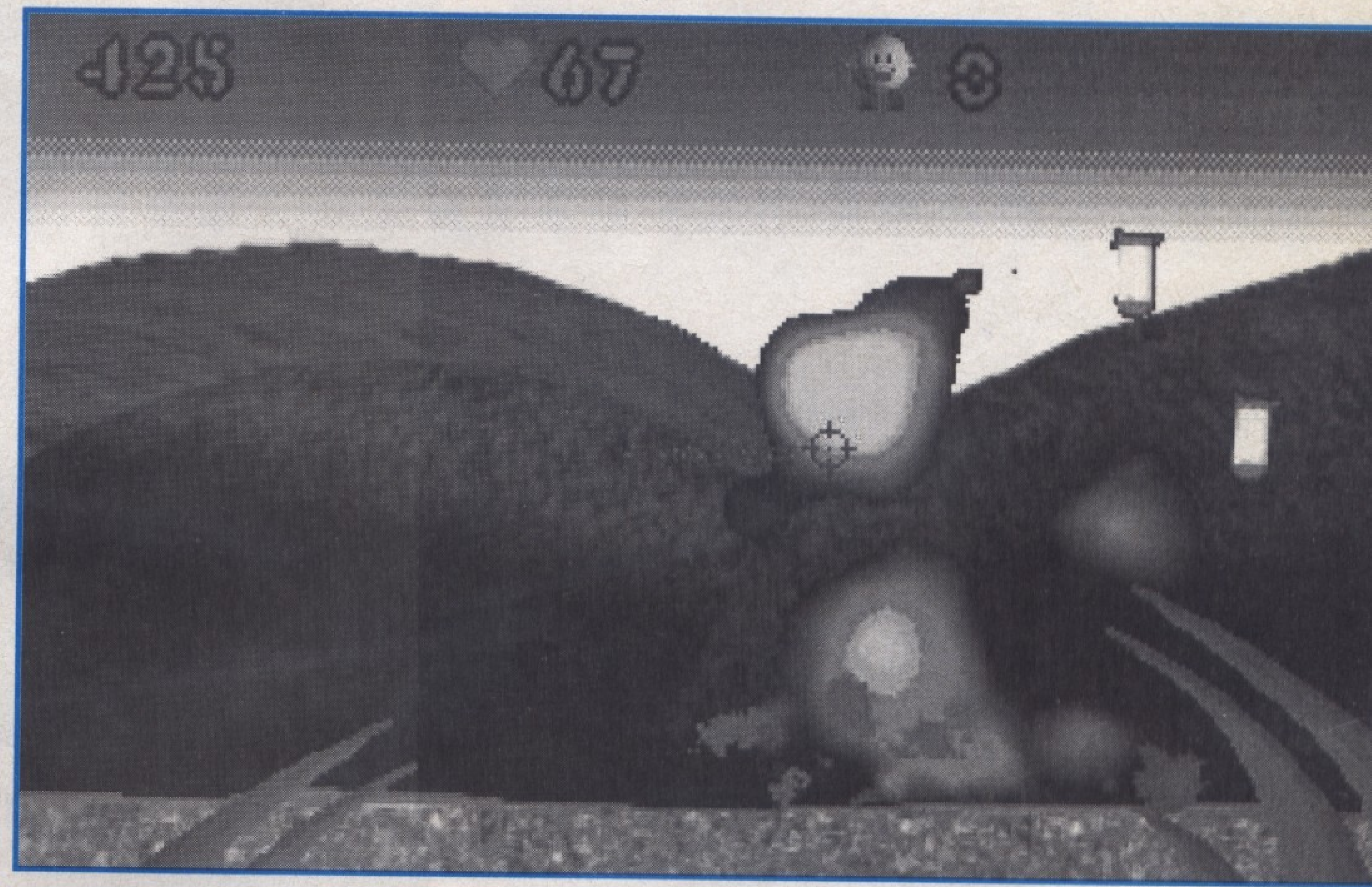
Enemigos que aparecen en forma de bola con pinchos.

```
if(marca>1500);
if(marca<7500);
graph=154;
Y=-20;
REPEAT
X+=INC_X;
Y+=INC_Y;
FRAME;
graph++;
if(graph>163)graph=154;
end
if(collision(type suelo));break;
end
if(collision(type bomba)or
collision(type bomba2)or
collision(type bomba3)or
collision(type bomba4));break;
end
if(x<0);x=320;
end
if(x>320);x=0;
end
until(collision(type
mirilla)and(key(_space)))puntos+=
50;From GRAPH=22 TO 30;FRAME;
END
if(graph>24)
sound(s_explo,40,160);
end
end
end
end
process suelo();
```

Ahora crea el suelo mediante un breve listado que nos será de gran utilidad.

```
begin
graph=15;
y=200;
x=160;
loop frame;
if(marca>40000);y=400;
end
end
end
process heli();
```

Los efectos visuales son, al menos, curiosos.



El proceso de los helicópteros enemigos se logra como detallamos a continuación:

```
begin
graph=43;
y=-900;
x=220;
repeat frame;
y=y+3;
if(y>150);y=y-3;
end
graph++;
if(graph>62)graph=43;
end
if(collision(type bomba)or
collision(type bomba2));break;
end
if(marca>2500);break;
end
until(collision(type
mirilla)and(key(_space)))puntos+=
100;from graph=63 to 71;frame;
END
if(graph>63)
sound(s_explo2,100,256);
end
end
process heli2();
```

El listado completo del juego, que han realizado estos aguerridos DIVmaniacos, es mucho más largo, y por obvios motivos de espacio no podemos incluirlos enteros, así como no aparece completa la explicación del juego. Sin embargo, quienes tengan interés en ver el código completo del juego, junto con algunas de las anotaciones del programador que nos ha obsequiado con este interesante programa, pueden asomarse al interior del CD-Rom que acompaña a la revista. En efecto, en él adjuntamos todo el texto, incluido el que ha aparecido en estas páginas. Si os ha gustado el juego, no dudéis en consultar este listado, pues aún faltan algunos de los procesos más interesantes del mismo, como el completo de los helicópteros enemigos.

Tus pequeños acompañantes iran alla donde tu vayas.



JUEGO DEL LECTOR: 2º

Autor: Álex Hernández Ortega

La limpieza sideral

Más allá del espacio sideral es el subtítulo que lleva este curioso programa, hecho a imagen y semejanza de los clásicos matamarcianos pero con toques de humor que amenizan nuestro viaje a través de la galaxia.

En este artículo os mostramos sólo el pequeño programa de presentación del juego. En el interior del CD-Rom incluimos el listado completo, muy interesante para ver cómo no se deben hacer algunas cosas. En efecto, según palabras del programador, "cuando empecé a programar el juego, en realidad se trataba de una prueba para ver las prestaciones de DIV, de hecho aprendí a programar en DIV creando este juego). Así que veréis que existe bastante código que podríamos considerar como 'paja', (por ejemplo, los *process* que contienen 'códigos ocultos'). Además, no está depurado, como los menús, que utilizan un *process* para cada uno, cuando se podría haber usado uno para todos e ir cambiando los parámetros."

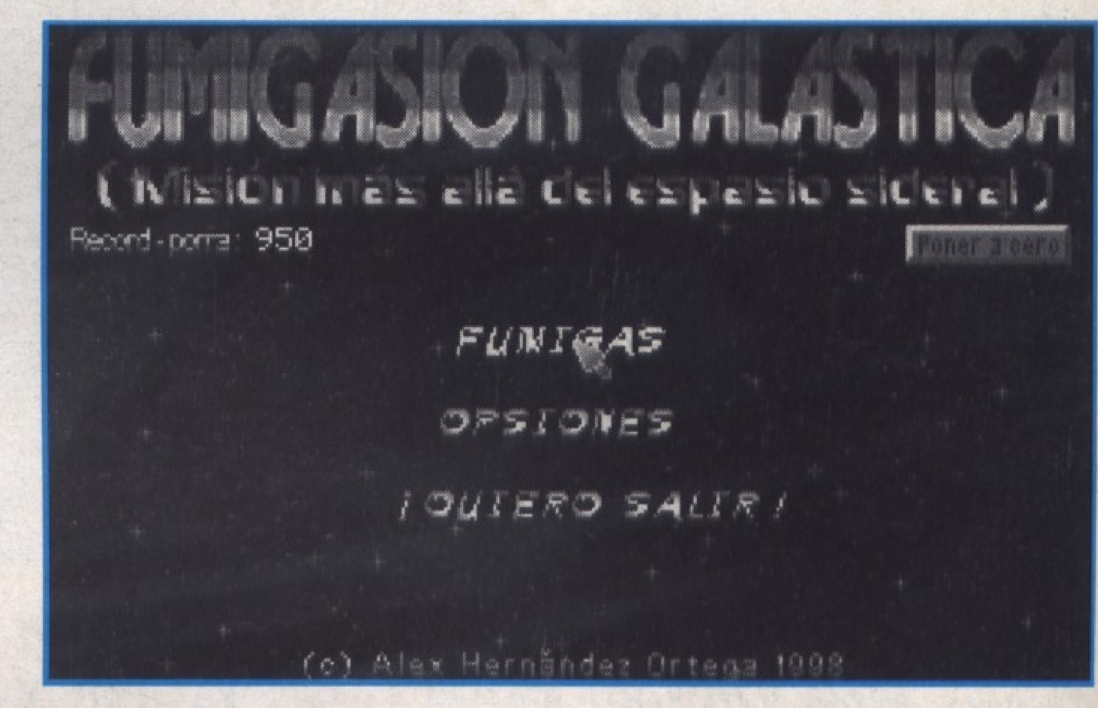
Los matamarcianos uno de los géneros más antiguo con más adeptos

También os preguntaréis por qué realizó un programa para los títulos y otro para el resto. Esto se debe a que cada vez que introducía un CD de audio en la unidad, tenía que reiniciar el programa. De este modo sólo se reiniciaba el segundo programa, no el primero.
Decíamos que había cosas que no estaban bien hechas, y había que tener en cuenta que el programador estaba aprendiendo cuando hizo esto. Sin embargo, muchas otras sirven perfectamente para mostrar las posibilidades de Div con los juegos de este tipo. Así pues, no perdáis detalle de este interesante listado. Lo hemos dividido en distintos grupos separados por un espacio mayor, de modo que queda claro a qué pertenece cada cosa.

```
PROGRAM soplagaitas_productions;

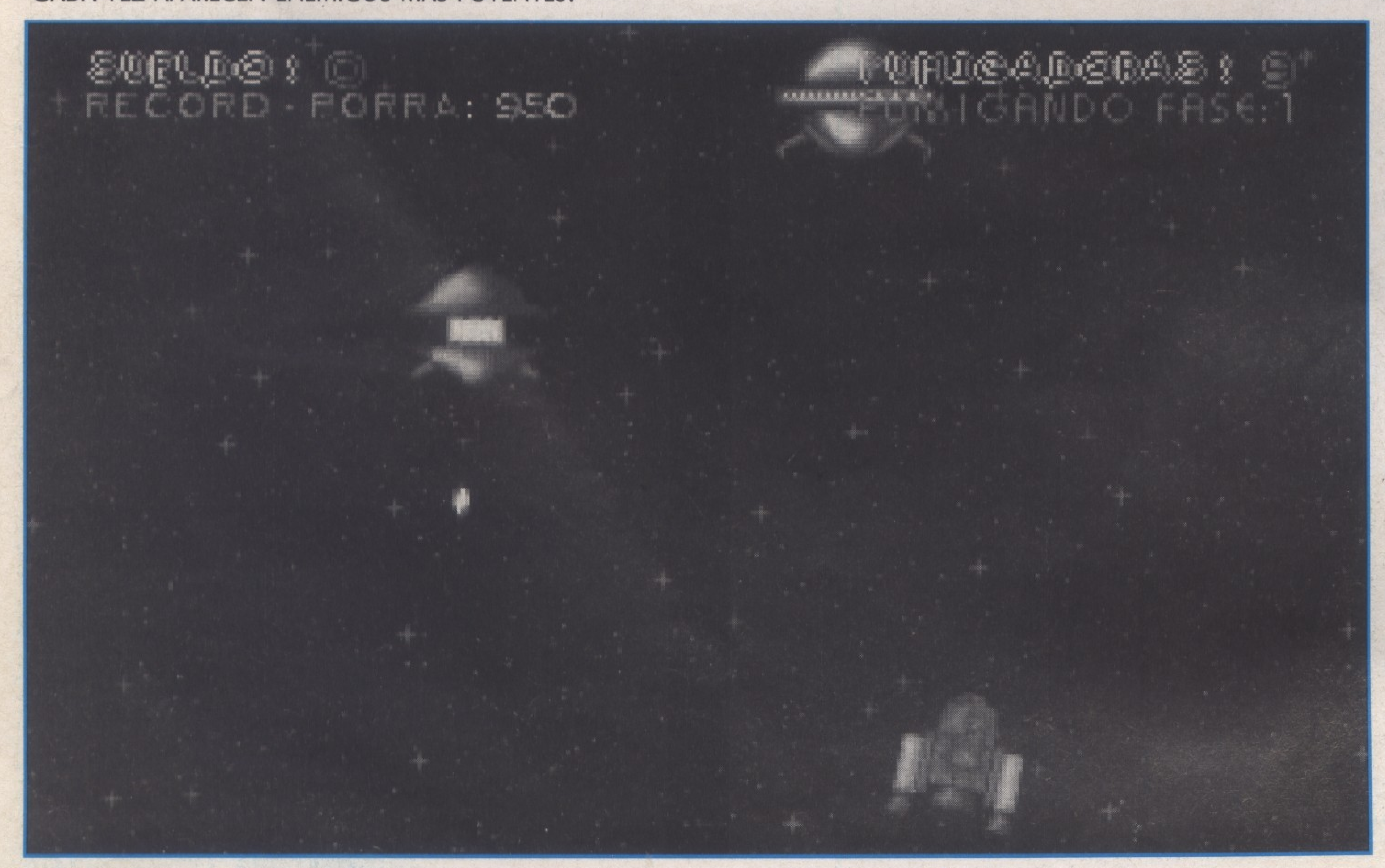
GLOBAL
sonido;
tiempo;
tiempo_2;
encendido;
BEGIN
  set_fps(24,0);
  setup.sound_fx=13;
  setup.master=13;
  set_volume();
  sonido=load_pcm("dat.040",0);
  load_fpg("dat.059");
  fade_off();
  alex_yo();
END
```

TODO UN DERROCHE DE SIMPATIA Y CREATIVIDAD.

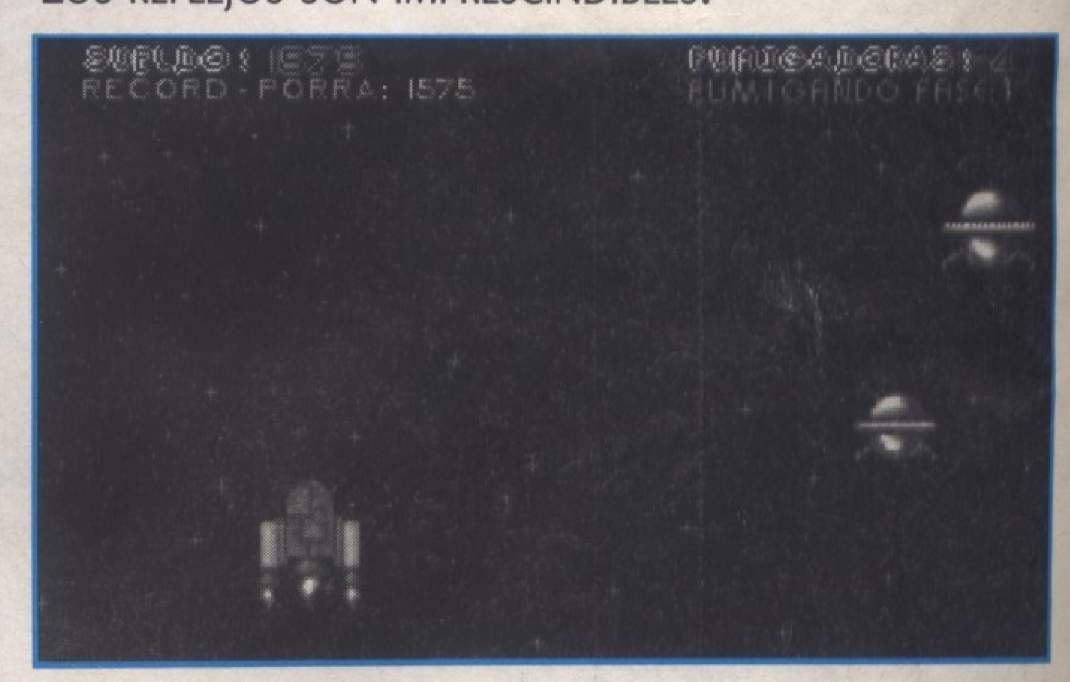


El juego está programado en DIV y todavía tiene algunos fallos

```
PROCESS alex_yo();
BEGIN
  scroll_fondo();
  LOOP
    IF(key(_a))
      BREAK;
    END
```



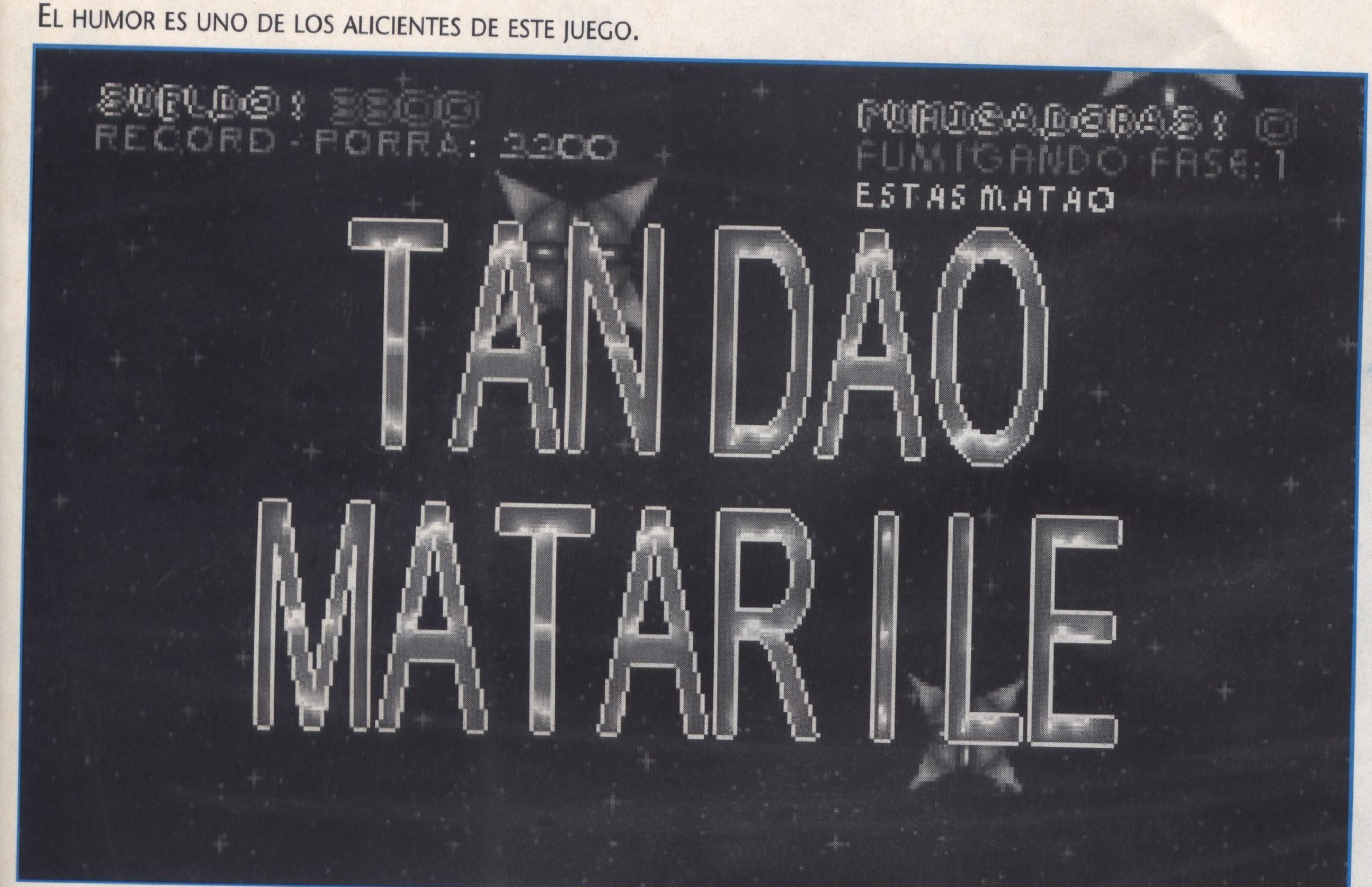
CADA VEZ APARECEN ENEMIGOS MAS POTENTES.



LOS REFLEJOS SON IMPRESCINDIBLES.

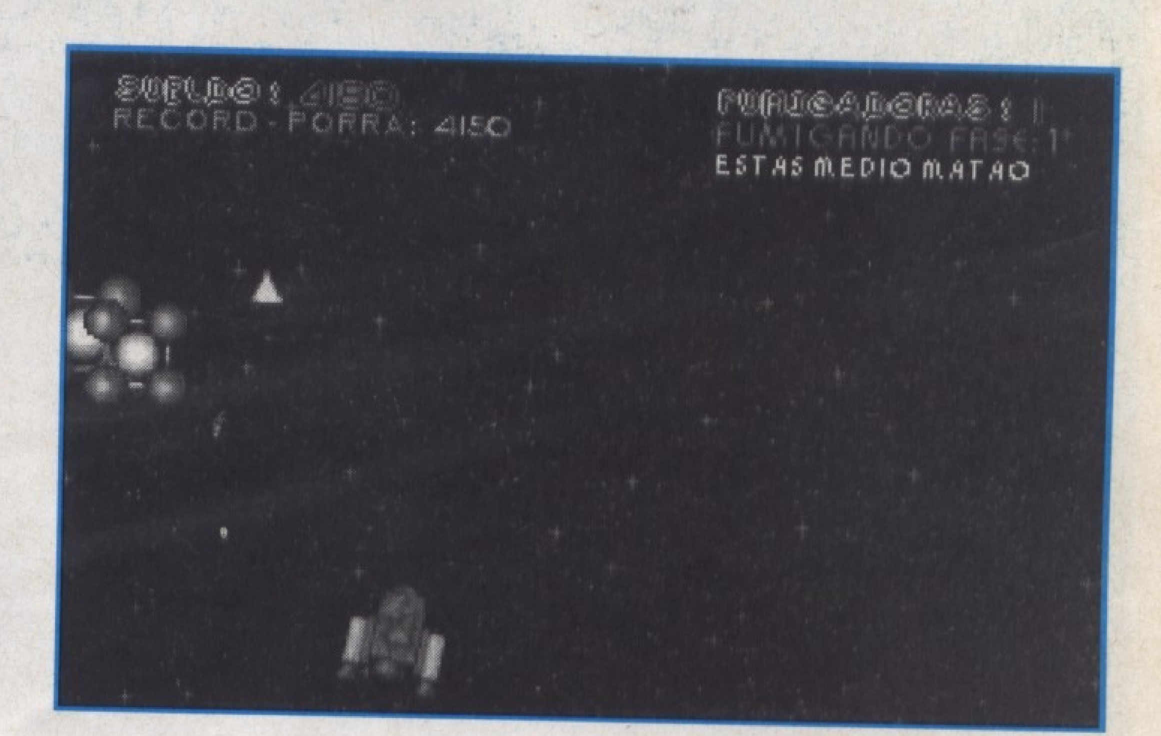


El humor es uno de los alicientes de este juego.



Las naves ocupan quiza demasiado espacio.






```
FRAME;
  END
  LOOP
    IF(key(_l))
      BREAK;
    END
  FRAME;
  END
  LOOP
    IF(key(_e))
      BREAK;
    END
  FRAME;
  END
  LOOP
    IF(key(_x))
      BREAK;
    END
  FRAME;
  END
  LOOP
```

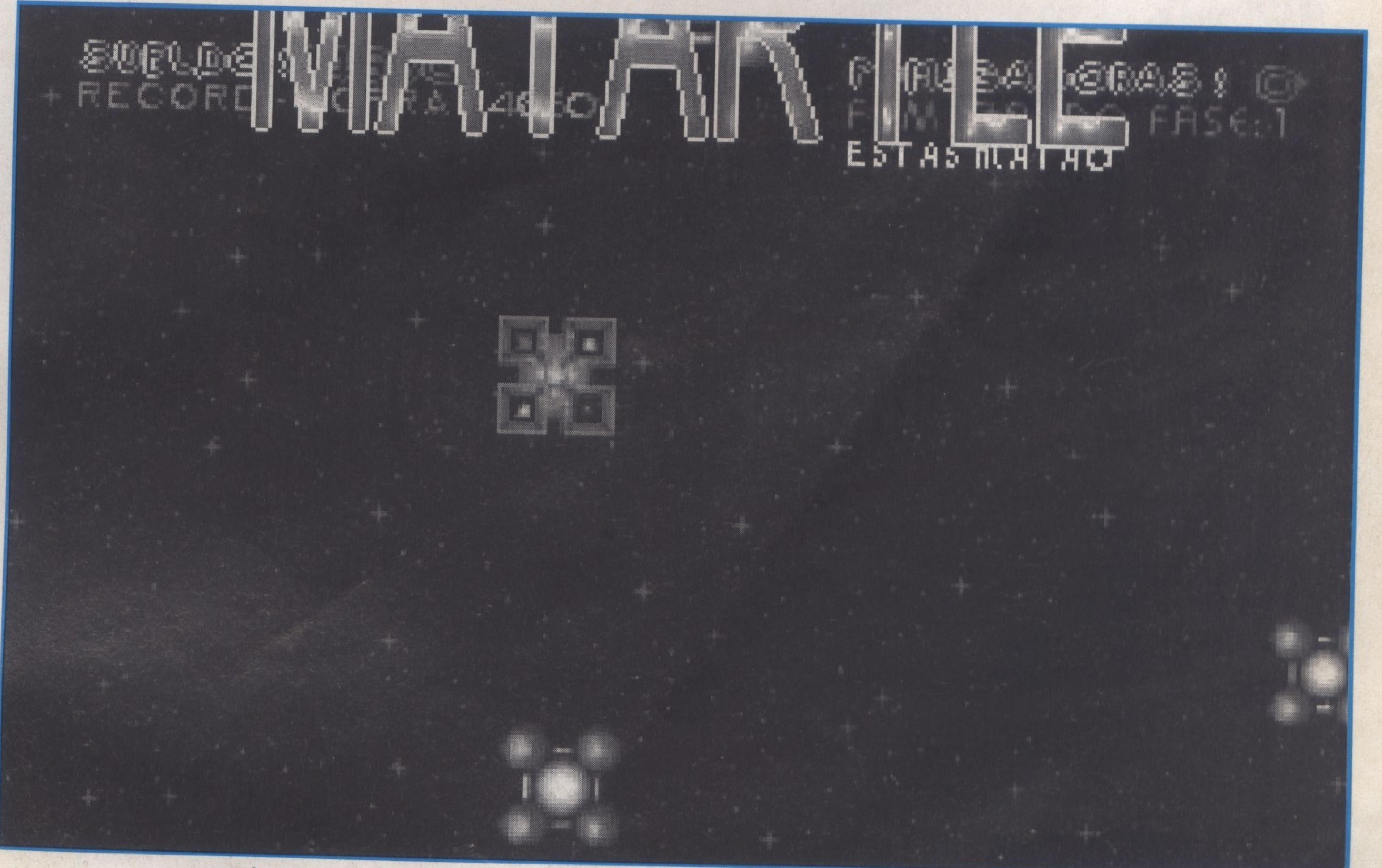
Fumigasión Galástica es un juego sencillo, pero con gran sentido del humor

```
    IF(key(_space))
      BREAK;
    END
  FRAME;
  END
  LOOP
    IF(key(_y))
      BREAK;
    END
  FRAME;
  END
```

```
  LOOP
    IF(key(_o))
      BREAK;
    END
  FRAME;
  END
  fade_off();
  let_me_alone();
  write(0,160,100,4,"(c) Alex Hern ndez Ortega
1998");
  LOOP
    scroll.x0=scroll.x0-1;
    scroll.y0=scroll.y0-1;
    tiempo_2=tiempo_2+1;

IF(encendido==0)fade_on();encendido=1;END
```

Nunca hay demasiados enemigos por pantalla.

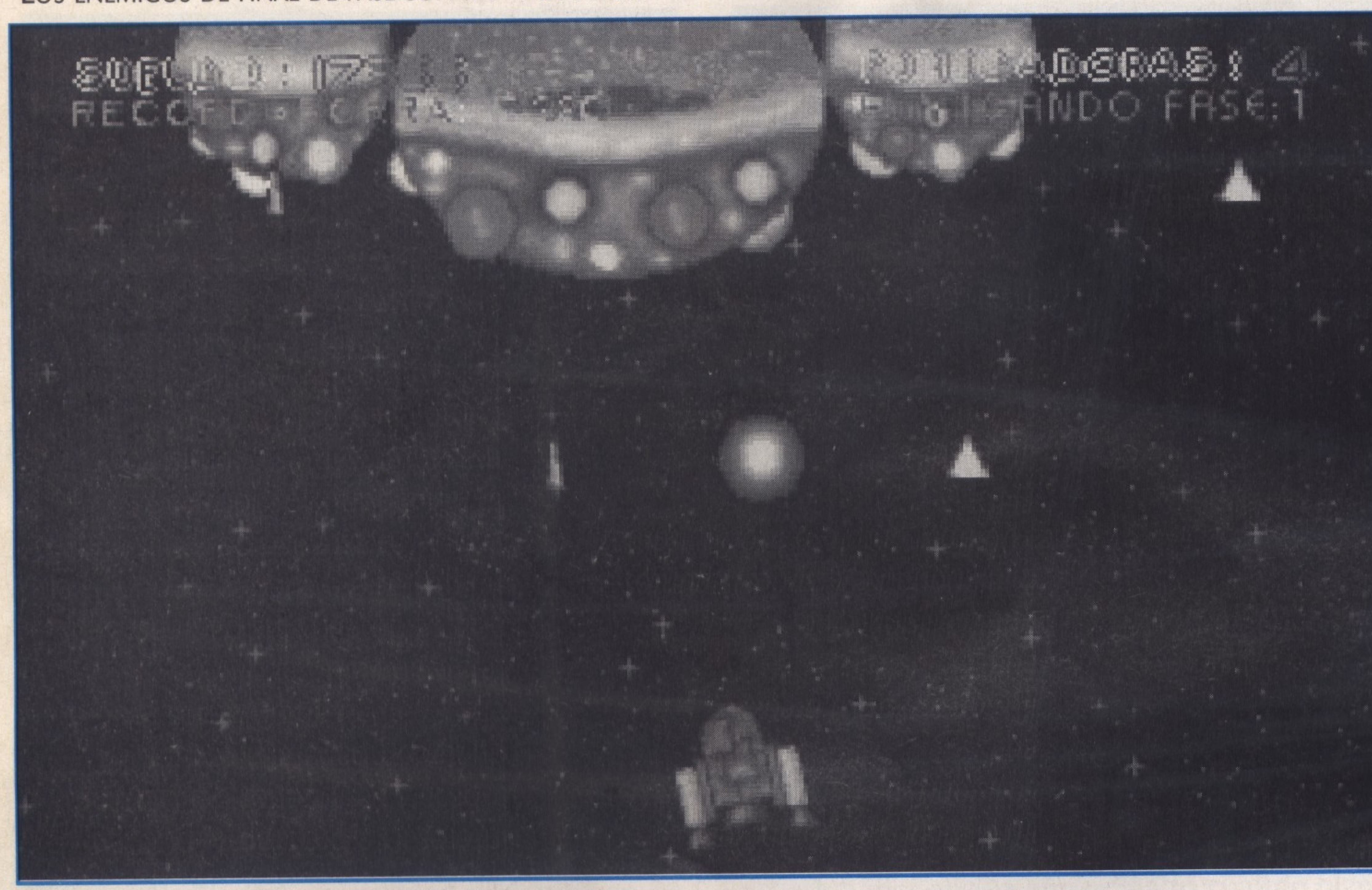



```
    IF(tiempo_2==250)BREAK;END
  FRAME;
  END
```

Una excelente muestra del sentido del humor hispano: simpatía, evidente y casposa

```
  fade_off();
  delete_text(all_text);
  encendido=0;
  tiempo_2=0;
  tiempo=0;
  alex_yo();
END
```

Los enemigos de final de fase son muy potentes.



```
PROCESS scroll_fondo();
BEGIN
   enciende();
   define_region(1,0,35,320,130);
   start_scroll(0,0,4,0,1,3);
   LOOP
      scroll.x0=scroll.x0+1;
      scroll.y0=scroll.y0+1;
   FRAME;
   END
END

PROCESS enciende();
BEGIN
   fade(100,100,100,1);
   WHILE(fading)
   FRAME;
   END
   sound(sonido,256,256);
   soplagaitas();
END

PROCESS soplagaitas();
BEGIN
   size=0;
   graph=1;
   x=160;
   y=72;
   LOOP
      size=size+5;
      IF(size==100)
         BREAK;
      END
   FRAME;
   END
   productions();
   LOOP
   FRAME;
   END
END

PROCESS productions();
BEGIN
   size=0;
   graph=2;
   x=160;
   y=102;
   LOOP
      size=size+5;
      IF(size==100)
         BREAK;
      END
   FRAME;
   END
   presenta();
   LOOP
   FRAME;
   END
END
PROCESS presenta();
BEGIN
   size=0;
   graph=3;
   x=160;
   y=135;
   LOOP
      size=size+2;
      IF(size==100)
         BREAK;
      END
   FRAME;
   END
   LOOP
      tiempo=tiempo+1;
      IF(tiempo==125)
         BREAK;
      END
   FRAME;
   END
```

Los enemigos de final de fase están realizados con buen gusto

```
   fade(0,0,0,3);
   WHILE(fading);
   FRAME;
   END
   exit("",0);
END
```

Atrevete a superar nuestro humilde record.

Te han despedio del curro
¡PERO HAS GANAO LA PORRA!
NUEVO RECORD.
21520

Autor: Carlos Pastor
rocky@primerama.es

Manga al servicio de DIV

El manga y el mundo de los videojuegos se han unido ya en más de una ocasión, en excelsos ejemplos como el de *Final Fantasy VII*. En esta ocasión ha sido uno de nuestros lectores quien se ha decantado por el cómic japonés para realizar un viedeojuego, esta vez por medio de DIV.

Desde que tuve mi primer ordenador (hace ya más de diez años) he estado esperando un programa como DIV. Un programa con el que pudiese entrar en el mundo de la programación de un modo sencillo y que me permitiera materializar un viejo sueño, crear juegos.
Cuando lo tuve entre mis manos decidí ponerme manos a la obra, e intentar realizar un juego sencillo para empezar a aprender, aunque no fuese nada del otro mundo.
Entonces empecé a hacer Manga Manía, y poco a poco fui descubriendo las grandes posibilidades que esconde DIV.

EL PROGRAMA PRINCIPAL

Después de haber declarado un montón de variables empieza el programa principal. Primero, fijo la resolución en 640x480 para poder visualizar correctamente los sugerentes fondos del juego. Luego sigo la velocidad en 200 fps para que el juego se desarrolle con soltura.
Y se leen el fichero de gráficos, los sonidos y los tipos de letras.
Echo esto se inicializa el proceso de presentación y se utiliza el bucle *while* para que el ordenador permanezca "quieto" hasta que el usuario pulse una tecla o pasen unos segundos.

Figura 5.



Cuando sale del bucle se inicializa el menú y se ponen unos cuantos *timer* a 0.

EL PROCESO PLAYER_1()

Primero ponemos el fondo y seleccionamos el gráfico del jugador. Luego empezamos un bucle *LOOP*.
Dentro del bucle le decimos al ordenador que la tecla *ESC* sirve para regresar al MENÚ, que la tecla *ENTER* sirve para activar al jugador para pintar y que *SPACE* es para desactivarlo.
La función *COLLISION()* me sirvió para comprobar que ningún enemigo nos tocó. Si nos toca se resta una vida al jugador, se le desactiva y se le pone en la posición inicial.
Entonces comprobamos si alguna de las teclas del cursor está pulsada y si el jugador está activado. Si se dan las dos condiciones se avanza el jugador en la dirección adecuada y se pinta un trozo de pantalla justo detrás suyo usando la magnífica función *MAP_BLOCK_COPY*.

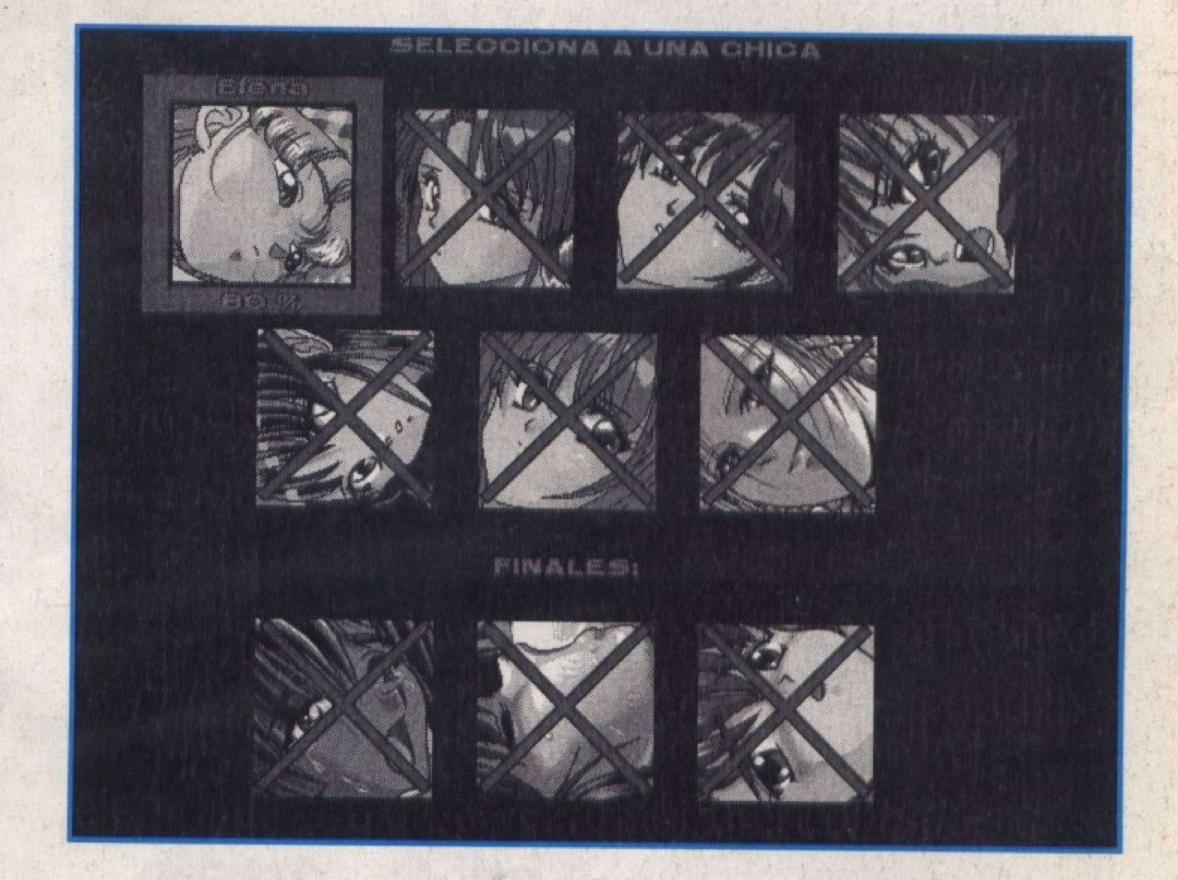
He estado esperando un programa como DIV. Cuando lo tuve, decidí ponerme manos a la obra

También se cuentan los píxeles pintados, para saber el tanto por ciento que el jugador ha conseguido. Si el jugador pasa por un sitio que ya ha sido pintado no se hace esta cuenta.
Y por último se activa el proceso llamado *MONOCROM()* si el tanto por ciento es mayor o igual a ochenta.

EL PROCESO MONOCROM()

Éste se encarga de pasar de la imagen en blanco y negro a la imagen en color.
Para conseguirlo usé el bucle *FOR* y de nuevo la función *MAP_BLOCK_COPY*.
Lo que hice fue repetir 200 veces la acción de poner un trozo en color de fondo, que cada vez se hacía más grande.

Figura 1.



También aprovecho para poner a *ELENA=FALSE*. De este modo ya no podré elegirla la próxima vez. Pongo el *contador_de_pixels=0* e inicializo el proceso de *puntuación()*.

EL PROCESO VIDAS()

Sirve, como no, para controlar las vidas del jugador.
Son una serie de 4 *if* que están dentro de otro *if*.
Si el jugador tiene menos de tres vidas entonces hace lo siguiente:

- Si las vidas son 2 y la variable *echo_una_vez* es igual a *FALSE* pone en la parte inferior izquierda de la pantalla el gráfico que enseña 2 corazones y pone *echo_una_vez=TRUE* para que el ordenador no esté continuamente poniendo este gráfico.
- Si las vidas son 1 y la variable *echo_una_vez2=FALSE* hace lo mismo con el gráfico que enseña 1 corazón.
- Si las vidas son 0 y *echo_una_vez3=FALSE* hace lo mismo.

Figura 3.

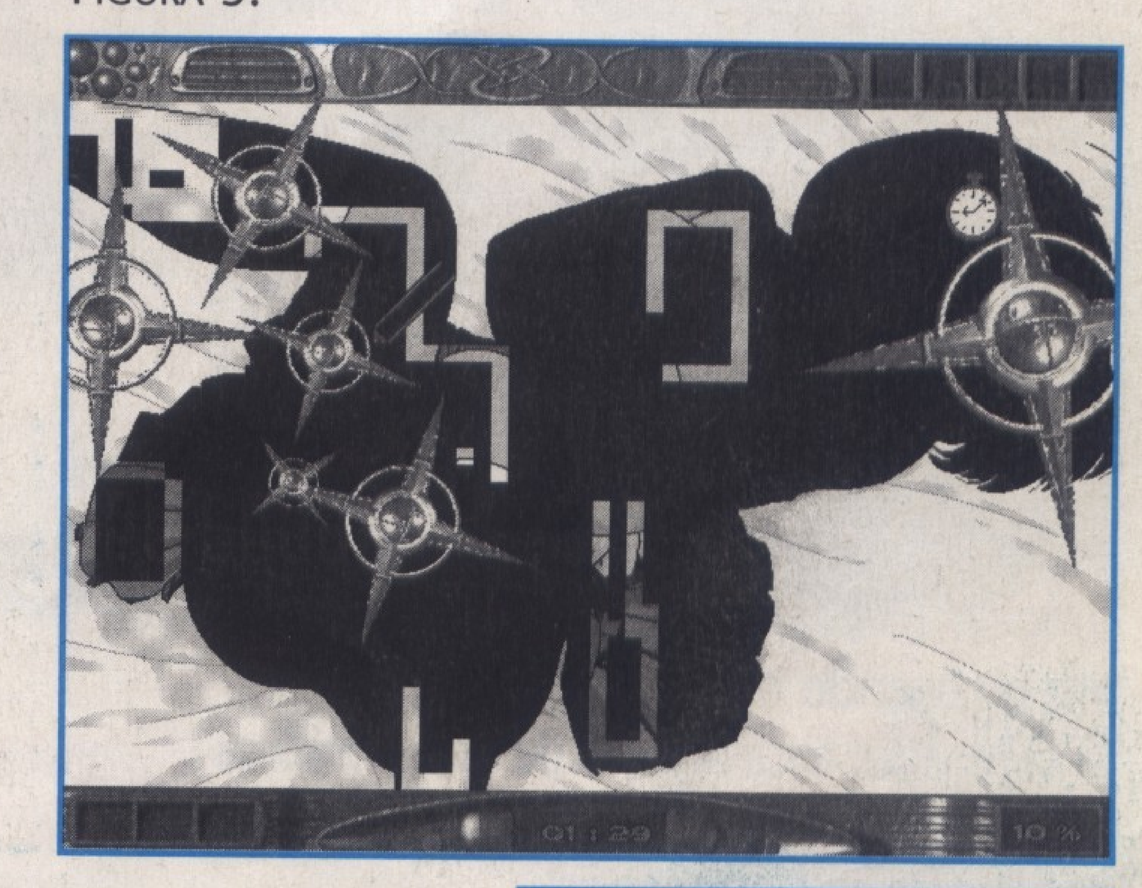


FIGURA 2.

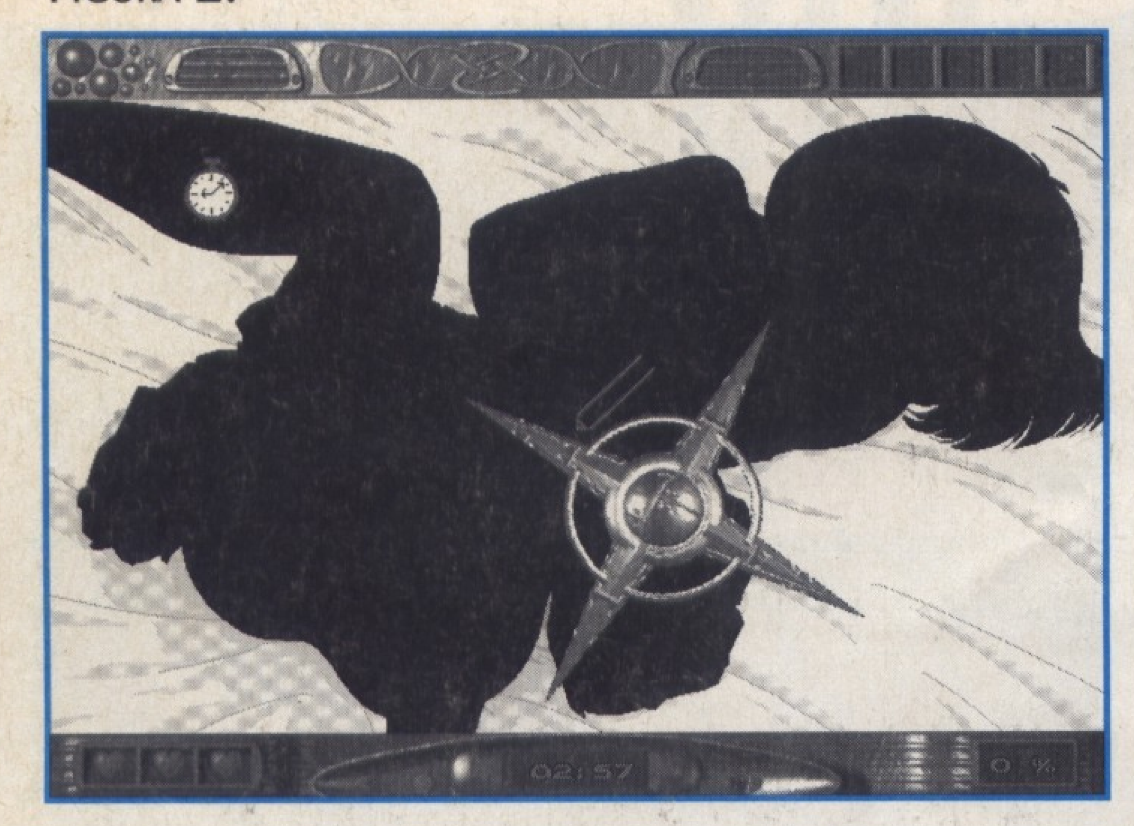


FIGURA 6.



Y si las vidas del jugador son menos de 0 entonces empieza el proceso *game_over()* que finaliza la partida.

EL PROCESO ENEMIGO()

Este proceso se encarga de mover el gráfico del enemigo por la pantalla.
Para ello seleccionamos el gráfico, el tamaño con *SIZE=60* y la posición. Luego, gracias a la función *ANGLE* hacemos que el gráfico vaya rotando.
Para conseguir que el enemigo rebote en la pantalla usé un viejo programa de una pelota que hice hace tiempo en C.
Basta con ver este ejemplo para explicarlo:

```
If(x>600)
        t_der=1;
        t_izq=0;
end
```

Así, si la posición de la coordenada *X* es mayor de 600, asignamos a la variable *t_der* el valor *1*. Esto significa que el enemigo ha tocado el borde derecho de la pantalla.
Y si miramos un poco más adelante en el código nos encontramos con lo siguiente:

```
If(t_der==1)
        pos_x_enemigo=pos_x_enemigo-1;
end
```

Con ello nos aseguramos que el enemigo nunca saldrá de la pantalla. Lo mismo sucede con los otros tres límites de la pantalla (arriba, abajo e izquierda).
Los demás procesos de los enemigos son idénticos excepto en pequeños detalles como el tamaño o el ángulo de rotación.

EL PROCESO ITEMS()

Aquí usé la función *RAND()* para que la posición de los items cambiara aleatoriamente, y la función *TIMER[]* para hacer que apareciesen de vez en cuando durante unos segundos.
Si el jugador toca alguno de estos gráficos (*COLLISION()*) suena un pitido y, según el gráfico que sea, aumenta el tiempo, disminuyen las vidas o se incrementan los puntos.

EL PROCESO MENÚ()

Es un proceso muy simple.
Si pulsamos el cursor hacia arriba se incrementa *contador_menu* en una unidad, y si pulsamos el cursor hacia abajo se resta uno.
Si, por ejemplo, hay un uno, se pone un gráfico iluminado en la primera opción y gráficos oscuros en las demás. Y así sucesivamente.

16 procesos, más o menos sencillos, para realizar un videojuego por medio de DIV

Si pulsamos *ENTER* se activa la opción que esté iluminada.

EL PROCESO SALIR()

Tampoco tiene ningún secreto. Aparece un gráfico preguntando si queremos salir. Si pulsamos *s* se activa la función *EXIT()*.
Para el efecto de hacerse grande y pequeño usé la función *SIZE*:

```
While(size<70)
        Size=size+7;
        Frame;
End
```

PROCESO CRÉDITOS() Y REGLAS DE JUEGO()

Simplemente se muestran los créditos en movimiento hasta que se pulsa una tecla.
En la opción de reglas de juego se muestra el texto explicativo hasta que se pulsa una tecla.
Muy sencillo.

PROCESO PUNTUACIÓN()

Primero se va sumando el porcentaje conseguido, luego los segundos que han quedado y por último las vidas. Cuando éstas llegan a 0 empieza el proceso *fin_demo()* y se termina la DEMO.

PROCESO RELOJ()

Se restan los minutos y segundos del reloj de la pantalla. Si se termina empieza de nuevo, pero se resta una vida.

PROCESO ELECCIÓN()

Largo pero sencillo.
Cuando se pulsa *ENTER* se borra la pantalla y empieza el proceso *preparado_ya()* y poco después *player1()*, seguido de *items()*, *reloj()*, *vidas()*, *tanto_por_ciento()* y *enemigo()*.

PROCESOS PRESENTACIÓN(), GAME_OVER(), PREPARADO_YA()

Son muy sencillos. Todos tienen en común que utilizan la función *SIZE()* para conseguir que se agranden o se hagan más pequeños.

PROCESO FIN_DEMO()

El último proceso. Primero puse en *arrays* varias frases curiosas. Luego, hice que apareciera el *FIN DEMO* durante unos segundos para más tarde dejar paso a los créditos.
Una vez que éstos se han ido de la pantalla, van saliendo aleatoriamente (gracias a *RAND()*) las frases curiosas hasta que se pulse una tecla.
Entonces se activa *EXIT()* y se termina el juego.

FIGURA 4.



PRÓXIMOS PROYECTOS

Ahora que creo haber aprendido algunas cosas sobre DIV es probable que me ponga con un proyecto realmente ambicioso. Se trata de un juego de acción con perspectiva cenital muy al estilo de *METAL GEAR SOLID*, pero sin punto de comparación claro (¡qué más quisiera yo!).
Lo que me gustaría captar en este futuro proyecto no es la gran calidad de imagen y sonido de *METAL GEAR SOLID*, sino su jugabilidad, la sensación de poder ser cogido "con las manos en la masa" en cualquier momento. En dos palabras: LA TENSIÓN.

Espero encontrar a un buen grafista que me ayude en los gráficos y las animaciones, aunque también estaría bien un músico para las melodías de fondo. Pero bueno, ya veremos...

PROGRAMAR JUEGOS ES COSA DE NIÑOS SI TE SUSCRIBES

a Div Manía

Si deseas estar en la vanguardia del mundo de la informática, suscribirse a DIV MANÍA es un primer paso acertado porque...

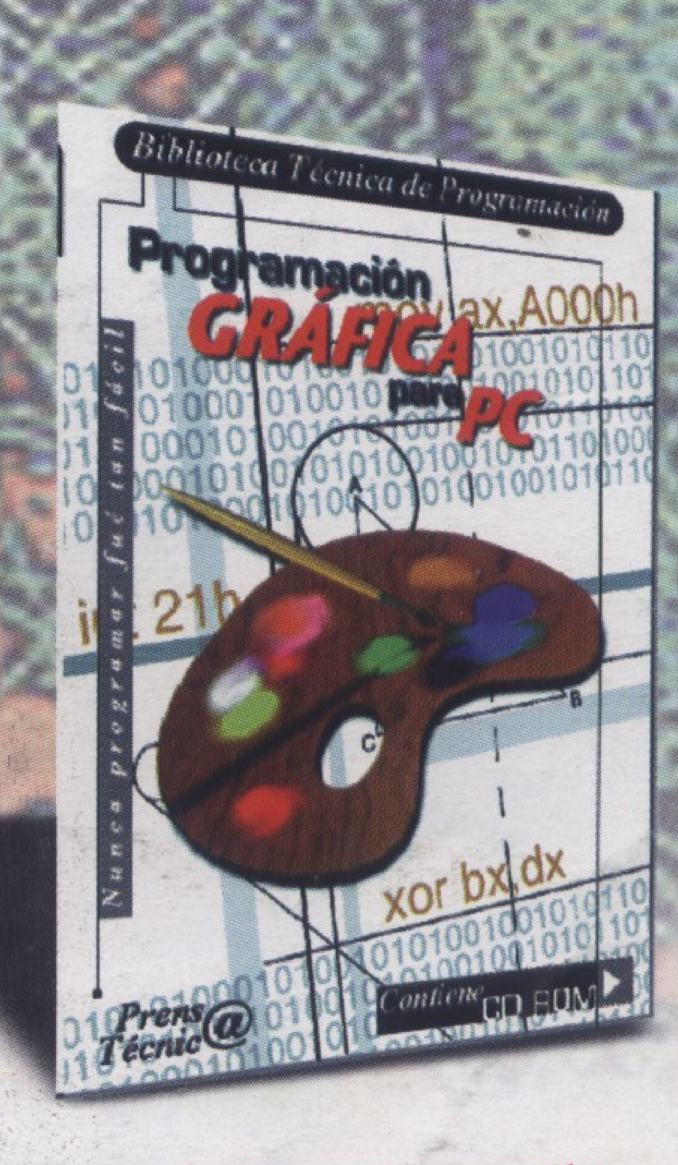
- *Es la única revista escrita por y para los programadores de videojuegos. Nuestra redacción está compuesta por veteranos desarrolladores, expertos del entorno DIV, grafistas y muchos otros profesionales del software de entretenimiento que dan lo mejor de sí a los lectores.*
- *Te ofrece lo último en el delicado campo de la programación de videojuegos, con los títulos que se encuentran en proceso y los productos recién salidos del horno o de los PCs.*
- *Para seguir avanzando hay que saber echar la vista atrás y a la vez no olvidarse del futuro, y DIV Manía empieza un nuevo camino.*
- *Nuestra revista presenta un look muy cercano a sus lectores, salido de las inquietudes de todos vosotros.*
- *Nunca nadie te ha ofrecido tanto por tan poco; nunca has tenido tan cerca la oportunidad de estar al día de lo último en programación por el mínimo esfuerzo de acercarte al quiosco o enviar nuestro cupón y recibir la revista en casa puntualmente cada dos meses.*
- *En el interior del CD-Rom, encontrarás los elementos con los que todos los programadores sueñan.*
- *Somos como tú y conocemos, más o menos, qué se esconde dentro de tu cabeza. Y si no lo conocemos aún, lo aprenderemos gracias a ti.*
- *Ofrecemos las más diversas sorpresas, las más interesantes ofertas, para que nunca olvides que la programación siempre está viva.*

Además el suscriptor tiene derecho a la siguiente oferta:

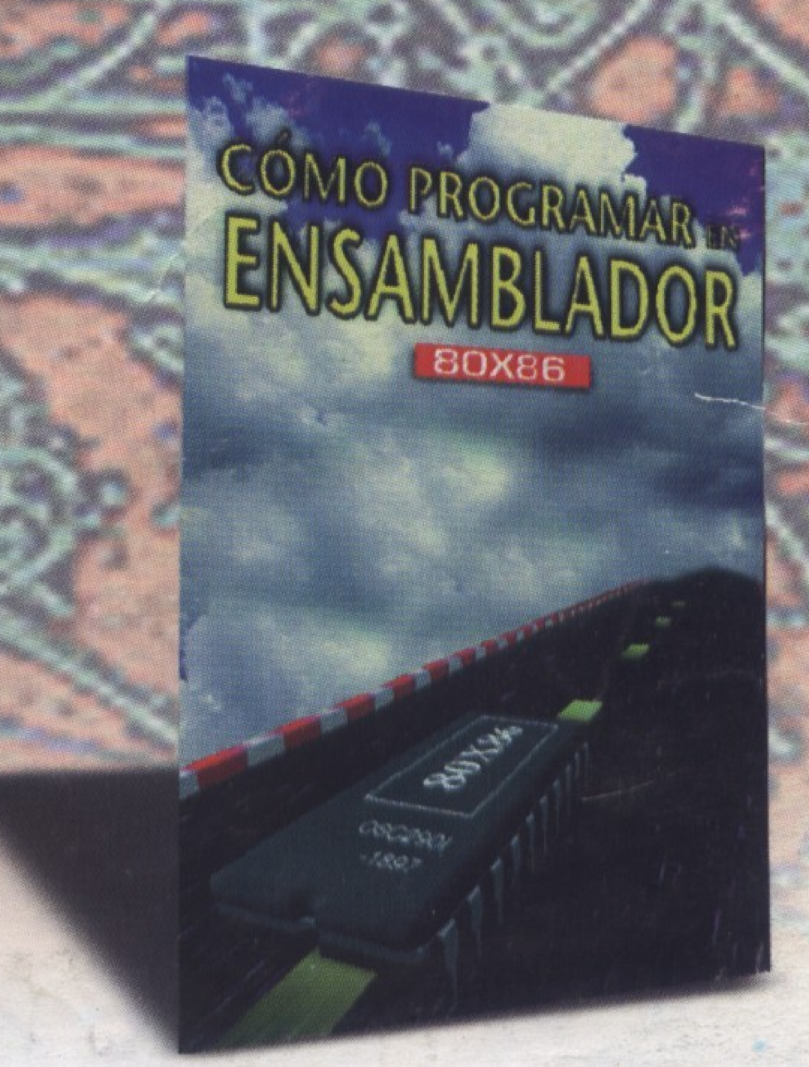
- **Con un año de suscripción (seis números) regalamos un producto a elegir entre:**
  "Programación Gráfica para PC"
  "Cómo programar en ensamblador"
- **Con dos años de suscripción (doce números) regalamos DIV 2.**



DIV II
Creación de juegos



PROGRAMACIÓN GRÁFICA PARA PC
Libro Programación



CÓMO PROGRAMAR EN ENSAMBLADOR
Libro Programación

Prensa Técnica se reserva el derecho de cambiar cualquiera de los regalos por otros de igual valor, en caso de agotar existencias.

CONTENIDO DEL CD ROM

DEMO DIV GAMES STUDIO 2

Con este nuevo CD os facilitamos una demo del esperado Div Games Studio 2, creación de Hammer Technologies. A pesar de ser una demo, podrás ejecutarla y programar con ella aunque con algunas limitaciones.

MACEDONIA MAGAZINE

En esta publicación digital que te ofrecemos con el CD ROM encontrarás un cantidad de información sobre cualquier tema que te interese: arte, juegos, programación, literatura y un largo etcétera; además de incluir links a otras páginas de Internet.

En este número no nos hemos olvidado de los Browsers y os ofrecemos los principales navegadores que hay ahora en el mercado: Netscape Comunicator 4.5 e Internet Explorer 4.01. Además de DIRECTX 6.0, la última versión de este novedoso programa.

DIV manía
NÚMERO 2
DEPOSITO LEGAL: M-42077-1998
VERSION ESPECIAL
Demo Div Games Studio 2
NOVEDAD
Macedonia magazine
LOS JUEGOS GANADORES
Comando pelota • Fumigasión Galástica • Manga Paradise
Además
Librerías
Fuentes • texturas • sonidos
Browsers
Netscape Comunicator 4.5
Internet Explorer 4.01
Utilidades
ACD See 2.3 • Panda Platinium • A. Reader
Graphics Workshop
P.S.P. 5.01 • Sea 1.3 • Virusscan 4.02 •
Winzip 7 • Cooledit
PC CD rom
Prens@ Técnic

LIBRERÍAS

Incluimos varias librerías que iremos ampliando poco a poco y que esperamos que sean de mucha utilidad para aquellos que hayan decidido internarse en el fascinante mundo de la programación de videojuegos. Te ofrecemos nuevos sonidos, fuentes y texturas.

Otra de las muchas cosas que podrás encontrar en nuestro CD: ACD SEE 2.3, un visor gráfico de gran rapidez; ANTIVIRUS PANDA PLATINUM, que detecta todos los virus que conoce y los nuevos; ACROBAT READER, un lector de ficheros con formato PDF; GRAPHICS WORKSHOP, otro visor gráfico que incluye un gran número de opciones; PAINT SHOP PRO 5.01, un programa de retoque fotográfico; SEA GRAPHICS VIEWER 1.3, el mejor visor gráfico para MSDOS; VIRUSSCAN DE MCAFEE 4.02, el antivirus más extendido; WINZIP 7, la última versión del compilador más conocido; COOLEDIT, uno de los más conocidos y el mejor editor de sonido.

Los programas de los lectores que han ganado nuestro concurso: Comando Pelota, un arcade divertido, Fumigasión Galástica un paranoico matamarcianos y Manga Paradise, un puzzle.

DIV GAMES STUDIO 2 Os ofrecemos en exclusiva la demo de segunda edición de DIV.



LIBRERÍAS Sonidos, fuentes y texto... Las mejores librerías a vuestro servicio.



PROGRAMA DEL LECTOR Los ganadores en el concurso realizado entre los lectores.

